Creating a Geodemographic Classification Using K-means Clustering in Python

Oliver Lock

University of New South Wales

Programmable Cities, 2020

Please note, this tutorial is adopted for the Sydney context and re-written for the Python programming language and is extensively based on work by Lansley & Cheshire in 2018. If you would like to do the same exercise later for the city of London, in R, see: https://data.cdrc.ac.uk/system/files/pdfgeodemographicslondongl.pdf

Introduction

> *"Geodemographic profiling (or 'geodemographics') is the neighbourhood scale analysis of people by where they live. Traditionally, it has fallen into the nomothetic approach to geographic enquiry - emphasizing the shared social, economic, and demographic characteristics of different types (or classes) of neighbourhoods, independent of their locations relative to unique places. As such, a type of neighbourhood may be widely scattered across a territory or jurisdiction. Geodemographic profiles thus provide summary indicators of the commonalities of social structure that link different locations." - Longley, 2017*

Geodemographic classifications group neighbourhoods (or sometimes even individual households) into types of similar characteristics based on a range of variables. They are a useful means on segmenting the population into distinctive groups in order to effectively channel resources. Such classifications have been effective deductive tools for marketing, retail and service planning industries due to the assumed association between geodemographics and behaviour.

For instance, typically a classification at the broadest level may distinguish cosmopolitan neighbourhoods with high proportions of young and newly qualified workers from suburban neighbourhoods with high proportions of settled families. Such classifications work because people of likeminded characteristics tend to cluster within cities. Whilst most geodemographic products are built within the commercial sector and sold by vendors, open source alternatives are available.

The following tutorial will provide you with the basic skills to build your own geodemographic classification using Python. All data and resources for this exercise are freely available.

Getting Started

In this exercise, we will classify areas based on their demographic characteristics using a form of statistical clustering known as k-means. The clustering will group cases based on their statistical similarity as exerted by the inputted variables. It is a very effective means of reducing large multivariate databases into a singular, but informative, indicator. Cases in the same group may be similar in terms of their geodemographic compositions, but they may not be geographically proximal as the technique is aspatial.

This exercise will take you through all the steps necessary to create a classification, from data selection, preparation, clustering and eventually interpretation. Although we have provided a pipeline of useful methods, you will see that the classification builder must make several analytical and subjective decisions in order to produce an optimum classification for a particular purpose. We have therefore provided some useful references at the end of this document for further reading. The first step to creating a geodemographic classification is considering what data to include and at what granularity Finer level data will allow you to capture more intricate variations and reduce any issues of ecological fallacy. However, we also require a good number of useful variables in order to effectively segment neighbourhoods. The following data has been prepared for this exercise:

Study area: Greater Sydney

Geography (spatial units): Statistical Areas 1 (SA1s)

Dataset: Australian Census, 2016

Statistical Area are spatially aggregate units created for the dissemination of 2016 Census data. SA1s are the lowest geographical level at which detailed census estimates are provided. SA1s generally have a population of 200 to 800 persons, and an average population of about 400 persons.

As they were built to be within predefined maximum and minimum population size thresholds, they are typically smaller in area in places where there are high population densities and much larger in sparsely populated areas.

The following datasets have been provided for you:

- Sydney-Census-Data.csv - a selection of Census variables at the SA1 level for Greater Sydney

A description of each variable can be found below. The variables contain the number of respondents who fit in this category, where the SA1 is their place of usual residence. Overall 45 variables are used to describe sociodemographic conditions of each SA1.

Column Name	Attribute descriptions
SA1_ID	Unique identifier for SA1
AGE_15_19 AGE_20_29 AGE_30_39 AGE_40_49 AGE_50_59 AGE_60_69 AGE_70_79 AGE_80_89 AGE_90_99 AGE_100_PLUS	Age brackets at 10-year intervals. Excludes those below 15.
MAR_NM MAR_W MAR_D MAR_S MAR_M	Marital Status Never married, Widowed, Divorced, Separated, Married
EDU_UNIV EDU_ALT_TERT EDU_SECONDARY	Education Level: University, Non-University Higher Education, Secondary School
INC_0_33 INC_33_64 INC 64_90 INC_90_150 INC_150_plus	Income per annum, e.g. income per annum between 33k to 63k.
LBS_YES LBS_NO	Labour force status: In the Labour Force (Employed full time or part time or looking for work) or Not in Labour Force.
IND_HTHEDU IND_KNOW IND_INDUST IND_POPSERVE	Industry of employment – divided from multiple categories in Census into 'Population-Serving' (retail & hospitality), 'Health & Education', 'Knowledge-Intensive' and 'Industrial'.
REL_BUD REL_CHR REL_HIN REL_ISL REL_JUD REL_OTH REL_SEC	Religious affiliation: Buddhism, Christianity, Hinduism, Islam, Judaism, Other, Secular (No religion)
COU_OCEANIA COU_NWEUROPE COU_SEEUROPE COU_NAFR_MEAST COU_SEASIA COU_NEASIA COU_SOUCENTASIA COU_AMERICAS COU_SUBSAHAFR	Region of birth: Oceania & Antarctica, North-West Europe, Southern and Eastern Europe, North Africa and Middle East, South-East Asia, North-East Asia, Southern and Central Asia, Americas, Sub-Saharan Africa
MTW_PT MTW_CAR MTW_ACTIVE MTW_OTHER MTW_NONE	Method of Travel To Work: Public Transport, Car, Active Transport, Other, none (Work from home / did or do not travel to work).

- SA_1_2016_Greater_Syd.shp - a shapefile of SA1s in Sydney

Load the data into Python

Download the data from our lesson materials and save them to a new folder on your computer. We will now load the census data into Python. We also need to set a working directory so that Python knows where to open and save data on your computer.

In this case, the new folder where you saved the data for this exercise will be the working directory.

```
import pandas as pd

import numpy as np

data = 'C:/…/ProgCities_Geodem_standardised.csv'

df = pd.read_csv(data)

df.head()
```

Load the data into Python

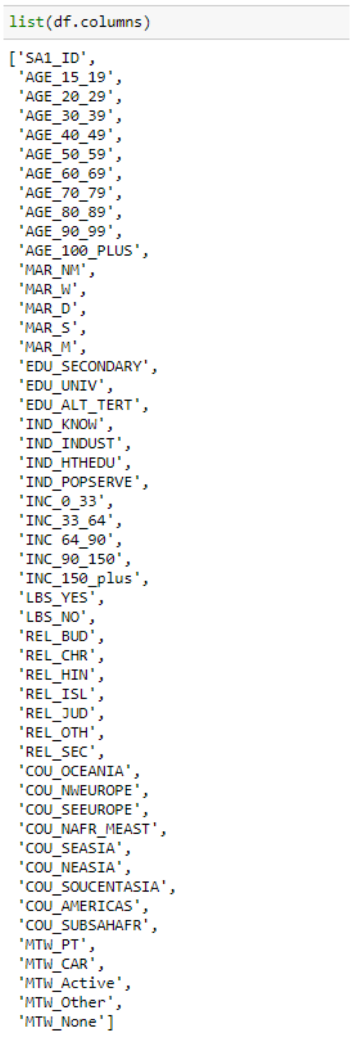
While many general purpose classifications use a wide range of variables on the population, those with a special application should be more selective. The most common uses of geodemographic classifications today are for marketing and service planning as generally consumer behaviour and service needs will vary by different geodemographic segments.

If we are classifying areas in the context of marketing we are interested developing a good understanding of residents' behaviour, their needs and their activities. Population data include several variables which can be used to provide useful inferences about consumer behaviour. For instance, areas with high proportions of families with young children will generally have different consumer traits to neighbourhoods with high proportions of young adults. Likewise, other domains such as socio-economics, ethnicity, tenure, car ownership, etc. . . will also influence consumer behaviour.

The aim here is to select a number of useful variables for your geodemographic classification from the 2016 Census data. The dataset we provided contains 45 variables. You might not need all of them. Indeed, having too many similar variables may skew the results of the classification toward particular domains or subgroups. Furthermore, some variables may be of little value due to the phenomena they offer. It will be worth considering what domain these variables represent before proceeding (E.g. age & sex, migration, family structure, ethnicity, employment & occupation, education and professional qualifications, housing, car ownership, health, mode of transport).

Data exploration

It might be worth conducting some basic data exploration. First, the list() function below will list the names of all of the variables in the dataset. Try also using the describe() function ; e.g. df.describe()

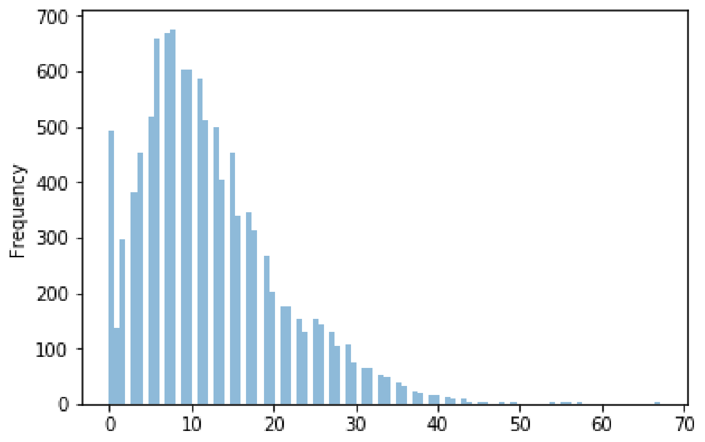


```
list(df.columns)
```

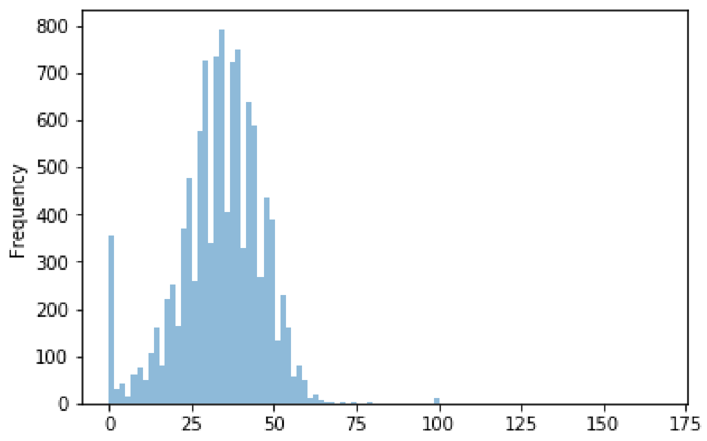
```
['SA1_ID',
 'AGE_15_19',
 'AGE_20_29',
 'AGE_30_39',
 'AGE_40_49',
 'AGE_50_59',
 'AGE_60_69',
 'AGE_70_79',
 'AGE_80_89',
 'AGE_90_99',
 'AGE_100_PLUS',
 'MAR_NM',
 'MAR_W',
 'MAR_D',
 'MAR_S',
 'MAR_M',
 'EDU_SECONDARY',
 'EDU_UNIV',
 'EDU_ALT_TERT',
 'IND_KNOW',
 'IND_INDUST',
 'IND_HTHEDU',
 'IND_POPSERVE',
 'INC_0_33',
 'INC_33_64',
 'INC 64_90',
 'INC_90_150',
 'INC_150_plus',
 'LBS_YES',
 'LBS_NO',
 'REL_BUD',
 'REL_CHR',
 'REL_HIN',
 'REL_ISL',
 'REL_JUD',
 'REL_OTH',
 'REL_SEC',
 'COU_OCEANIA',
 'COU_NWEUROPE',
 'COU_SEEUROPE',
 'COU_NAFR_MEAST',
 'COU_SEASIA',
 'COU_NEASIA',
 'COU_SOUCENTASIA',
 'COU_AMERICAS',
 'COU_SUBSAHAFR',
 'MTW_PT',
 'MTW_CAR',
 'MTW_Active',
 'MTW_Other',
 'MTW_None']
```

It is also useful to look at histograms of the distribution of your data.

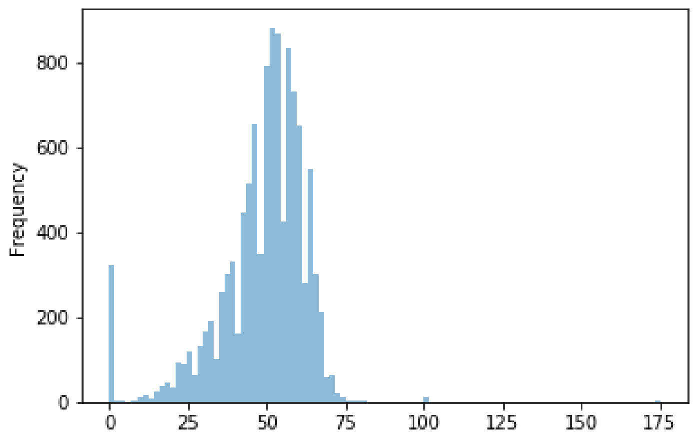
```
%matplotlib inline
ax = df['MTW_PT'].plot.hist(bins=100, alpha=0.5)
```



```
ax = df['MTW_CAR'].plot.hist(bins=100, alpha=0.5)
```



```
ax = df['MAR_M'].plot.hist(bins=100, alpha=0.5)
```



Subsetting data

When you have decided which variables you are interested in, you can create a new subsetted data object. To
select multiple columns in Python you can adapt the code below to include the variables of your liking. Here we
have selected 23 variables as an example.

```
subset_df=
df[['SA1_ID','AGE_30_39','MAR_M','IND_KNOW','IND_INDUST','IND_HTHEDU','IND_
POPSERVE','MTW_CAR','MTW_PT','INC_0_33','INC_90_150']]

subset_df.head()
```

Data standardisation

Standardisation between areas

To reduce the effects of unbalanced base population sizes across each of the small area units (Output Areas)
the variables all need to be transformed into percentages. Fortunately, this has already been done for you within the Sydney-Census-Data_normalised.csv file.

However, please note that if you were to create your own percentages from aggregate population data it is
important to ensure you are dividing the nominator by the correct dominator. The dominators create from Census data will vary between data tables (i.e. the total population, the total number of households, the
total of the economically active population, and so forth).

The vast majority of values in this data will be between 0 to 100%. However, when using Small Areas in the Australian Census, a tiny degree of random '1's are added into the dataset to preserve anonymity. For areas with very few people, these can give values over 100%. We have left these in for this exercise; however we recommend you also address these case by case in future, especially if you are seeing interesting results arising from these underpopulated areas.

Standardisation between variables

So that erratic values within variables do not inadvertently dominate the clustering process, the input variables need to be standardised so that they each contribute an equal weight. Standardising the data will
also make the final outputs much easier to interpret.

This can be achieved in various ways but here we will adopt a straightforward approach by calculating Z-scores for each variable. Z-scores (or standard scores) describe a standardisation format where all values are represented as the number of standard deviations from the mean (0). Therefore, positive Z-scores indicate that the values are above the mean.

Crucially, it is now easy to compare the variables on a common scale.

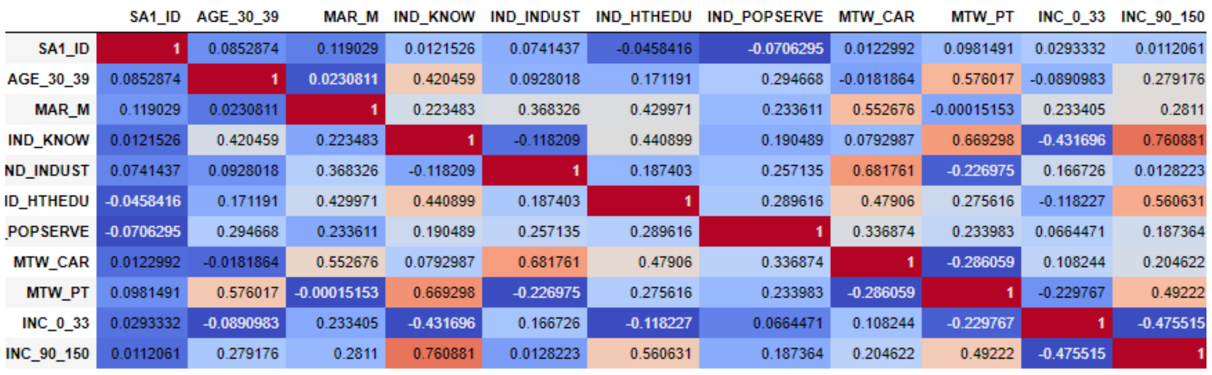
```
standardised_df = subset_df
cols = list(df.columns)
cols.remove('SA1_ID')
# now iterate over the remaining columns and create a new zscore column
for col in cols:
   standardised_df[col]          =          (standardised_df[col]          -
standardised_df[col].mean())/standardised_df[col].std(ddof=0)

standardised_df.head()
```

Remember: *z-score = (x- mean of the population)/standard deviation of the population*

So let's loop this code so it restandardises all of your variables and produces a new data frame called standardised. The i represents what is being looped, in this case, it loops through all of the columns in the subsetted data frame (remember in our example we did not include the "SA_ID" column from the original
data).

It is useful to test for associations between the final selection of variables. Variables that are collinear would essentially be conveying very similar distributions. This could give a particular phenomena a higher weighting in the final classification. To check for multicollinearity, create a Pearson's correlation matrix for the dataset. The following instructions will produce a Pearson's correlation matrix and accompanying significance table.

```
corr = standardised_df.corr()
corr.style.background_gradient(cmap='coolwarm')
```



	SA1_ID	AGE_30_39	MAR_M	IND_KNOW	IND_INDUST	IND_HTHEDU	IND_POPSERVE	MTW_CAR	MTW_PT	INC_0_33	INC_90_150
SA1_ID	1	0.0852874	0.119029	0.0121526	0.0741437	-0.0458416	-0.0706295	0.0122992	0.0981491	0.0293332	0.0112061
AGE_30_39	0.0852874	1	0.0230811	0.420459	0.0928018	0.171191	0.294668	-0.0181864	0.576017	-0.0890983	0.279176
MAR_M	0.119029	0.0230811	1	0.223483	0.368326	0.429971	0.233611	0.552676	-0.00015153	0.233405	0.2811
IND_KNOW	0.0121526	0.420459	0.223483	1	-0.118209	0.440899	0.190489	0.0792987	0.669298	-0.431696	0.760881
ND_INDUST	0.0741437	0.0928018	0.368326	-0.118209	1	0.187403	0.257135	0.681761	-0.226975	0.166726	0.0128223
ID_HTHEDU	-0.0458416	0.171191	0.429971	0.440899	0.187403	1	0.289616	0.47906	0.275616	-0.118227	0.560631
POPSERVE	-0.0706295	0.294668	0.233611	0.190489	0.257135	0.289616	1	0.336874	0.233983	0.0664471	0.187364
MTW_CAR	0.0122992	-0.0181864	0.552676	0.0792987	0.681761	0.47906	0.336874	1	-0.286059	0.108244	0.204622
MTW_PT	0.0981491	0.576017	-0.00015153	0.669298	-0.226975	0.275616	0.233983	-0.286059	1	-0.229767	0.49222
INC_0_33	0.0293332	-0.0890983	0.233405	-0.431696	0.166726	-0.118227	0.0664471	0.108244	-0.229767	1	-0.475515
INC_90_150	0.0112061	0.279176	0.2811	0.760881	0.0128223	0.560631	0.187364	0.204622	0.49222	-0.475515	1

For every pair of data, a Pearson's coefficient and significance value are calculated. A Pearson's coefficient (or r-square value) is presented on a scale between -1 and 1. The greater the value the greater the association between the two variables. The direction of the value equates to the direction of the relationship, negative
values represent negative relationships for instance.

Look at the correlation matrix, do the associations seem logical? As a rule of thumb, two variables with coefficients greater than ±0.8 can be considered to be highly correlated. In these cases, it might be reasonable to remove the variable which correlates the greatest with the other variables in the matrix so you are left with the most unique one. If you removed a variable, you may wish to find new variables to replace it. However, of course, some researchers might argue that it is reasonable to retain correlated variables so long as each of them as unique value and meaning. Indeed, so long as the correlations are not quite perfect (i.e 1 or -1), both variables in the pairs may contain useful outliers which could help distinguish certain area types.

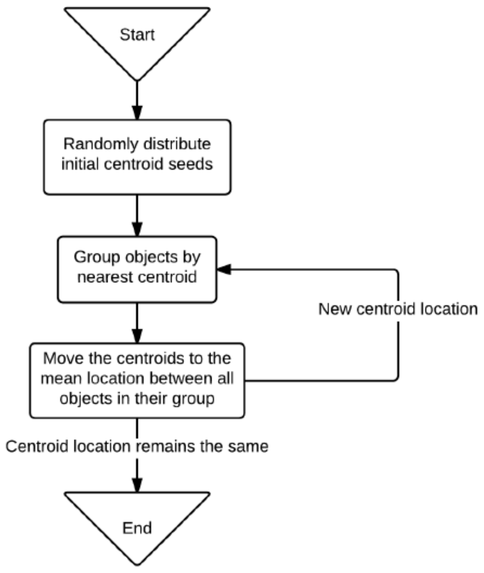
I've listed some examples of highly correlated variables. Refer to the data descriptions on previous pages, can you think of why these would be related?

AGE_80_89 and MAR_W
AGE_15_19 AND EDU_SECONDARY
COU_SOUCENTASIA AND REL_HINDU
IND_KNOW AND INC_90_150

Spend some time modifying your data subset selection and previous code until you are satisfied with the variables chosen before you move onto the next task.

K-means Clustering

We will use the k-means clustering method. It is a top-down approach whereby the number of cluster groups is predefined. K-means is an iterative relocation algorithm based on an error sum of squares measure. The algorithm seeks to reduce the sum distance between each data point and their respective cluster centre. The diagram below illustrates the basic algorithm process of k-means clustering. It starts by randomly allocating seeds across a multidimensional space as defined by the variables, each case is then assigned to the nearest seed centroid. In other words, the cases are assigned into cluster groups based on the seed they are nearest to across the multiple variables. Following the first iteration, a new seed is created at the centroid of each of the clusters. Each case is then re-assigned to clusters based on the distance to the nearest of these new centroids. This process repeats iteratively until the centroid seed locations cannot be moved as an optimum solution has been reached (See Harris *et al.*, 2005).

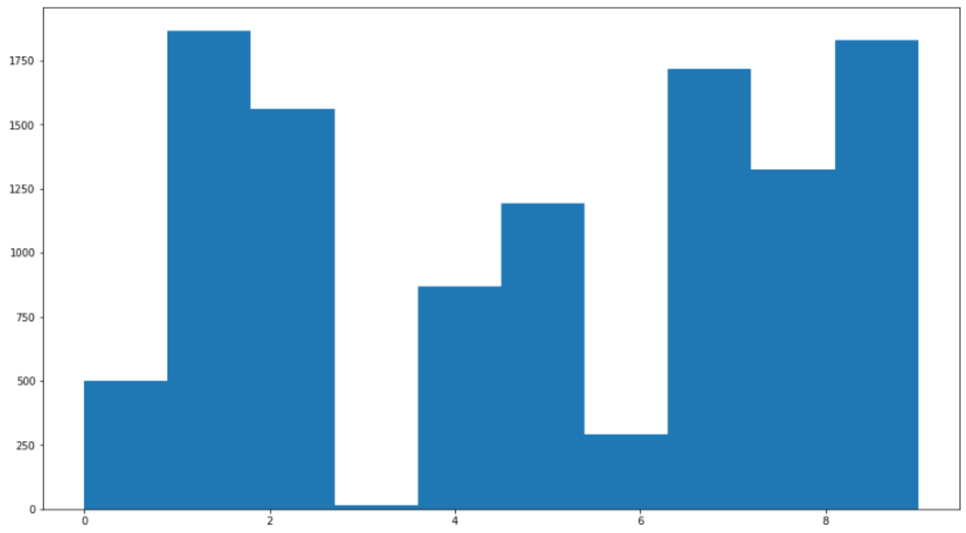

Start
Randomly distribute initial centroid seeds
Group objects by nearest centroid
New centroid location
Move the centroids to the mean location between all objects in their group
Centroid location remains the same
End

The process of the k-means algorithm (excerpted from Lansley et al, 2015). The code is annotated below. You can view the full parameters by opening scikit-learn's k-means documentation.

```
#We need to import the sklearn library
from sklearn.cluster import KMeans

#We need to change the format of the data including removing the ID and focus on key variables for analysis
k_means_array = standardised_df[['AGE_30_39','MAR_M','IND_KNOW','IND_INDUST','IND_HTHEDU','IND_POPSERVE','MTW_CAR','MTW_PT','INC_0_33','INC_90_150']]
k_means_array = k_means_array.as_matrix()

#To create a K-means cluster with 10 clusters, simply type the following script
#create Kmeans object and specify number of clusters
kmeans = KMeans(n_clusters=10)
## call the fit method on kmeans and pass the data you want to
kmeans = kmeans.fit(k_means_array)
## labels for your data points, with the number between 0 to 10
labels = kmeans.predict(k_means_array)
#location of final centres of each cluster
cluster_centres = kmeans.cluster_centers_
```

If this script has run, then you have run the clustering. If you would like to look at the core characteristics of each of the cluster groups, you can plot a histogram of the cluster labels.

```
plt.hist(labels)
```



We can see here that the cluster '3' has the lead, and cluster '10' has the highest number of SA1s.

What is the optimum number of clusters?

There is no right answer to this question. Even making judgements using some guidance on criteria involves
a level of subjectivity. Your task is now to choose an appropriate number of clusters for your geodemographic
classification.

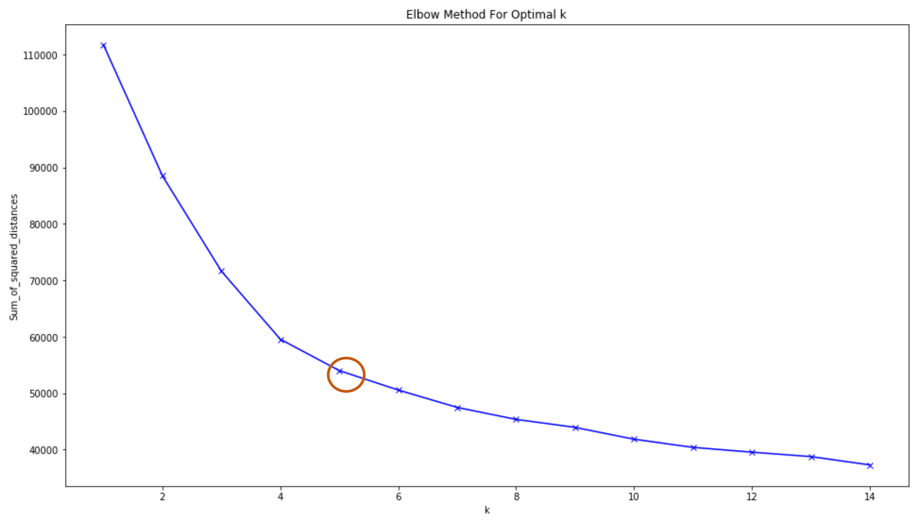
Aims of the cluster analysis:
• Each cluster should be homogeneous as possible
• Each cluster group should be distinct from the other groups
• The groups should be as evenly sized as possible

In addition, to each of these, we must also consider the compositions of the cluster groups. It is important that each of the characteristics of each cluster are distinguishable and relatable to real-life neighbourhood types. We will cover interpreting the cluster centres later in this tutorial.

Each cluster should be homogeneous as possible
Generally, the greater the number of clusters the closer the
each case is to their cluster centroid on average. However, of course, with more groups the classification becomes more difficult to interpret and the differences between some groups may become more subtle. A
measure we can use to check this is the within-cluster sum of squares which is the sum of the squared deviations from each observation to the cluster centroid. So larger sums indicate that the cluster is more dispersed and less homogenous.

A plot of the within groups sum of squares by number of clusters extracted can help determine the appropriate number of clusters (k). We graph the relationship between the number of clusters and Within Cluster Sum of Squares (WCSS) then we select the number of clusters where the change in WCSS begins to level off (called the elbow method).

```
Sum_of_squared_distances = []
K = range(1,15)
for k in K:
   km = KMeans(n_clusters=k)
   km = km.fit(k_means_array)
   Sum_of_squared_distances.append(km.inertia_)

plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```

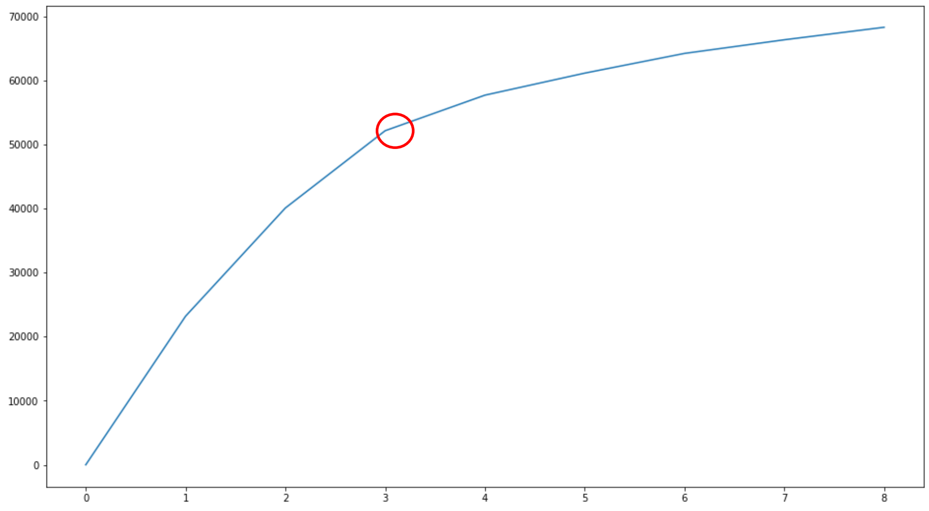


Here the ‘elbow’ of the arm is ***around 5***, which means we should try using a cluster number value close to 5. On some plots this is much more obvious than others.

Each cluster group should be distinct from the other groups
We can also measure how distinctive each of the clusters are in each model overall by looking at the between-cluster sum of squares. This value essentially measures how far apart clusters from different centres are. If the value is low then cases from different clusters will not be that distinctive. It is also important to observe the cluster centres to ensure that the compositions of each of the groups are logical and sufficiently unique. We can look at the between-cluster sum of squares to understand how discriminatory the models are when different numbers of groups (*k*) are produced.

This is not a readily available function; so we create an elbow() function in order to draw this one:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn
from sklearn.cluster import KMeans
import numpy as np
from scipy.spatial.distance import cdist, pdist

df=
standardised_df[['AGE_30_39','MAR_M','IND_KNOW','IND_INDUST','IND_HTHEDU','
IND_POPSERVE','MTW_CAR','MTW_PT','INC_0_33','INC_90_150']]

def elbow(df, n):
    kMeansVar = [KMeans(n_clusters=k).fit(df.values) for k in range(1, n)]
    centroids = [X.cluster_centers_ for X in kMeansVar]
    k_euclid = [cdist(df.values, cent) for cent in centroids]
    dist = [np.min(ke, axis=1) for ke in k_euclid]
    wcss = [sum(d**2) for d in dist]
    tss = sum(pdist(df.values)**2)/df.values.shape[0]
    bss = tss - wcss
    plt.plot(bss)
    plt.show()

elbow(df,10)
```



Using the two plots, is it possible to determine an appropriate number of groups?

There are also other methods for identifying statistically optimal groups such as the average silhouette method and gap statistic method.

The groups should be as evenly sized as possible

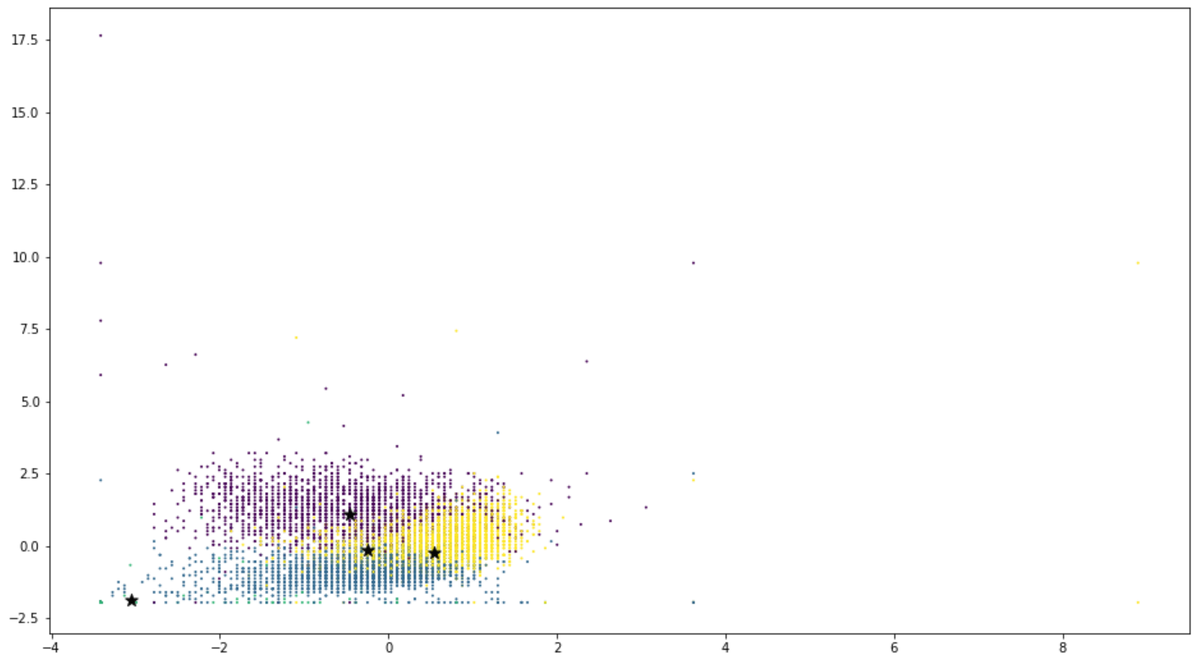
This is more of a rule of thumb for geodemographics. However, if a single cluster group comprises a large share of the data, then it is obviously undesirable for practitioners hoping to glean a better understanding of how the needs of neighbourhoods may vary.

You might notice that your K-means has produced a small cluster group which has exaggerated cluster centre values for a small number of variables, this is not unusual when attempting to segment social groups. There are two main ways to edit the size of the groups without manipulating the variables: joining similar
groups or splitting large groups. If one cluster group is particularly dominant in size, you can force it into two groups by isolating the cases (statistical areas) from that group only and running a 2 group solution k-means. With only one group selected, you can run a k-means as you did before, but change the number of groups produced to 2. As only one group is selected, the output areas from other groups will be excluded from the analysis. If you are happy with the results you now need to combine the two new groups with the original group membership variable.

Plotting your clusters – two dimensions

What are the X and Y axis on this plot?

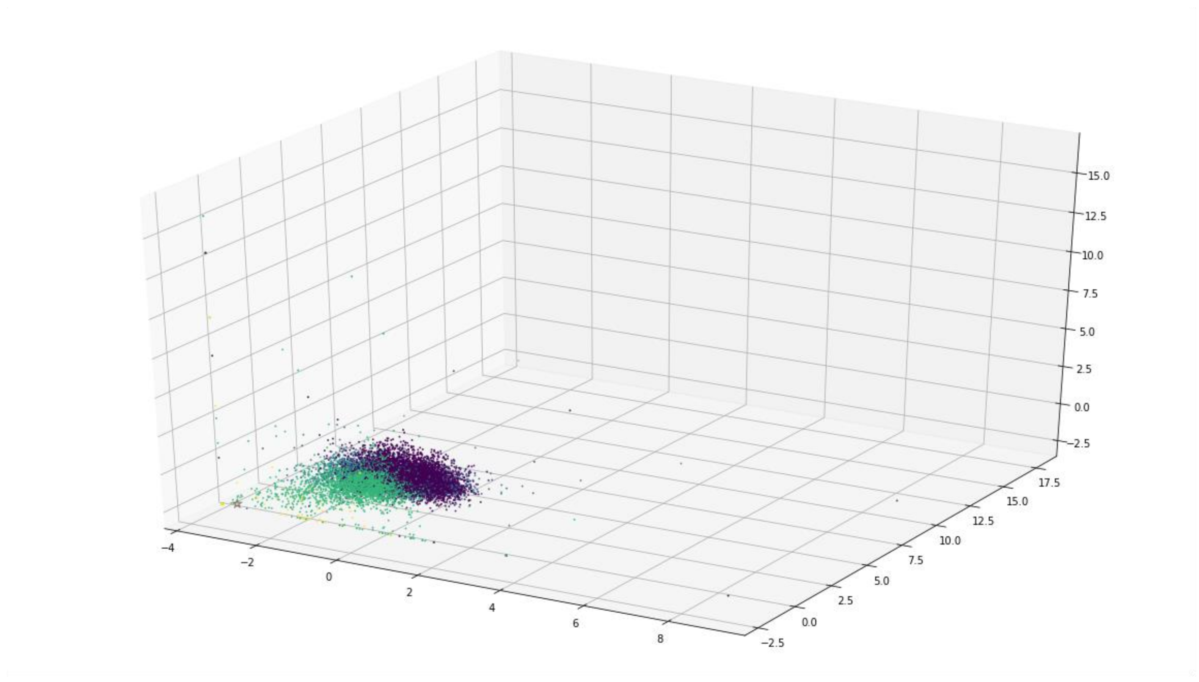
```
from sklearn.cluster import KMeans

k_means_array =
standardised_df[['AGE_30_39','MAR_M','IND_KNOW','IND_INDUST','IND_HTHEDU',
'IND_POPSERVE','MTW_CAR','MTW_PT','INC_0_33','INC_90_150']]
k_means_array = k_means_array.as_matrix()

kmeans = KMeans(n_clusters=4)
kmeans = kmeans.fit(k_means_array)
labels = kmeans.predict(k_means_array)
cluster_centres = kmeans.cluster_centers_


plt.rcParams['figure.figsize'] = (16,9)
fig,ax = plt.subplots()
ax.scatter(k_means_array[:,1],k_means_array[:,2],marker='o',s=1,c=labels)
ax.scatter(cluster_centres[:,1],cluster_centres[:,0], marker='*', c='#050505', s=100)
```



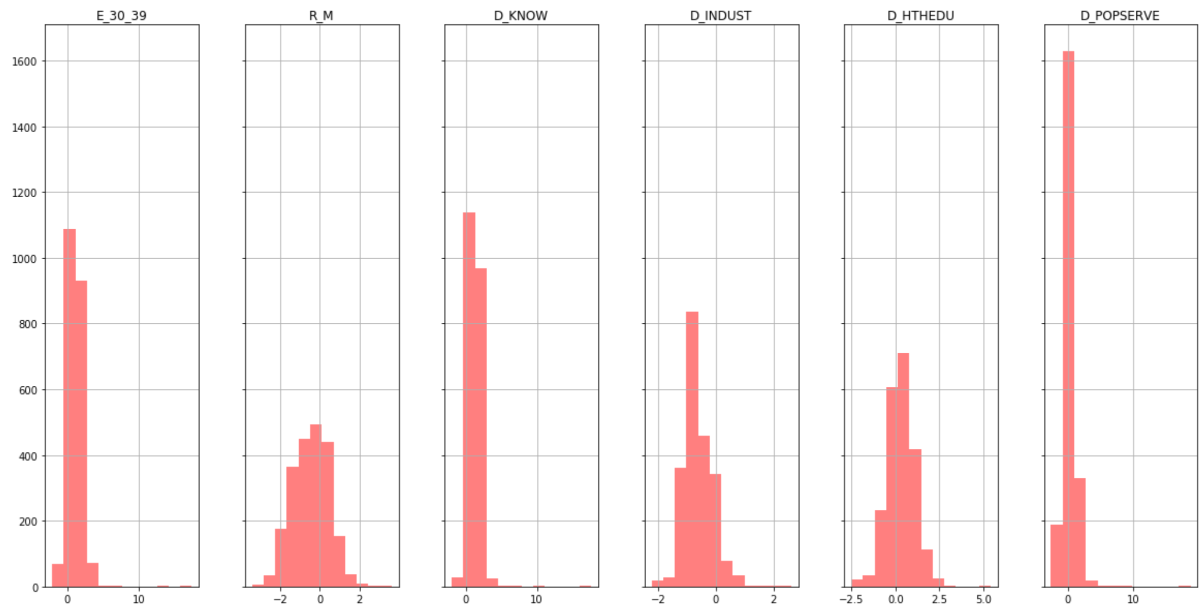
Plotting your clusters – three dimensions

What are the X and Y and Z axis on this plot?

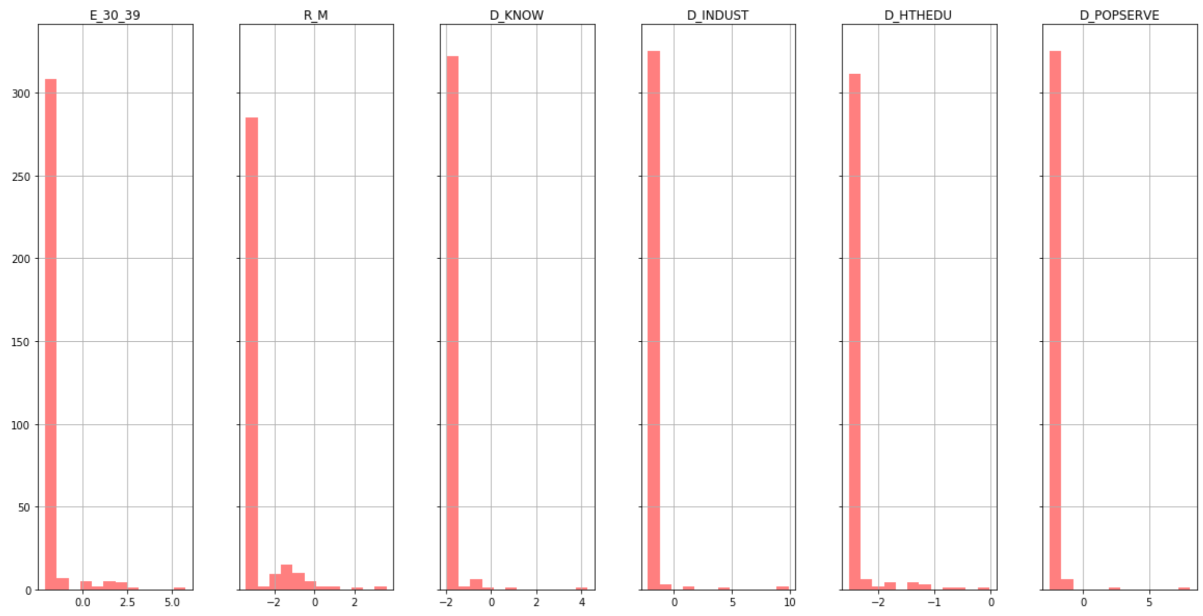
```
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize']=(16,9)
fig= plt.figure()
ax = Axes3D(fig)
ax.scatter(k_means_array[:,1],k_means_array[:,2],k_means_array[:,3],marker='o',s=1,c=labels)
ax.scatter(cluster_centres[:,1],cluster_centres[:,2],cluster_centres[:,3],          marker='*', c='#050505', s=100)
```



```
##Cluster Profile – generate histograms of the distribution of variables within each cluster


import seaborn as sns

subset_df['labels'] = labels

#subset_df=subset_df.drop(columns=['SA1_ID'])

# Change the value highlighted to alter the cluster selection

label_select = subset_df[subset_df['labels']==1]

fig, ax = plt.subplots(1, 6, sharex='col', sharey='row', figsize=(20, 10))

#number of cluster labels

n = 6

for j in range(n):

   label_select.hist(column=label_select.columns[j], bins=12, ax=ax[j], alpha=0.5, color='red')

   ax[j].set_title(label_select.columns[j][2:])
```



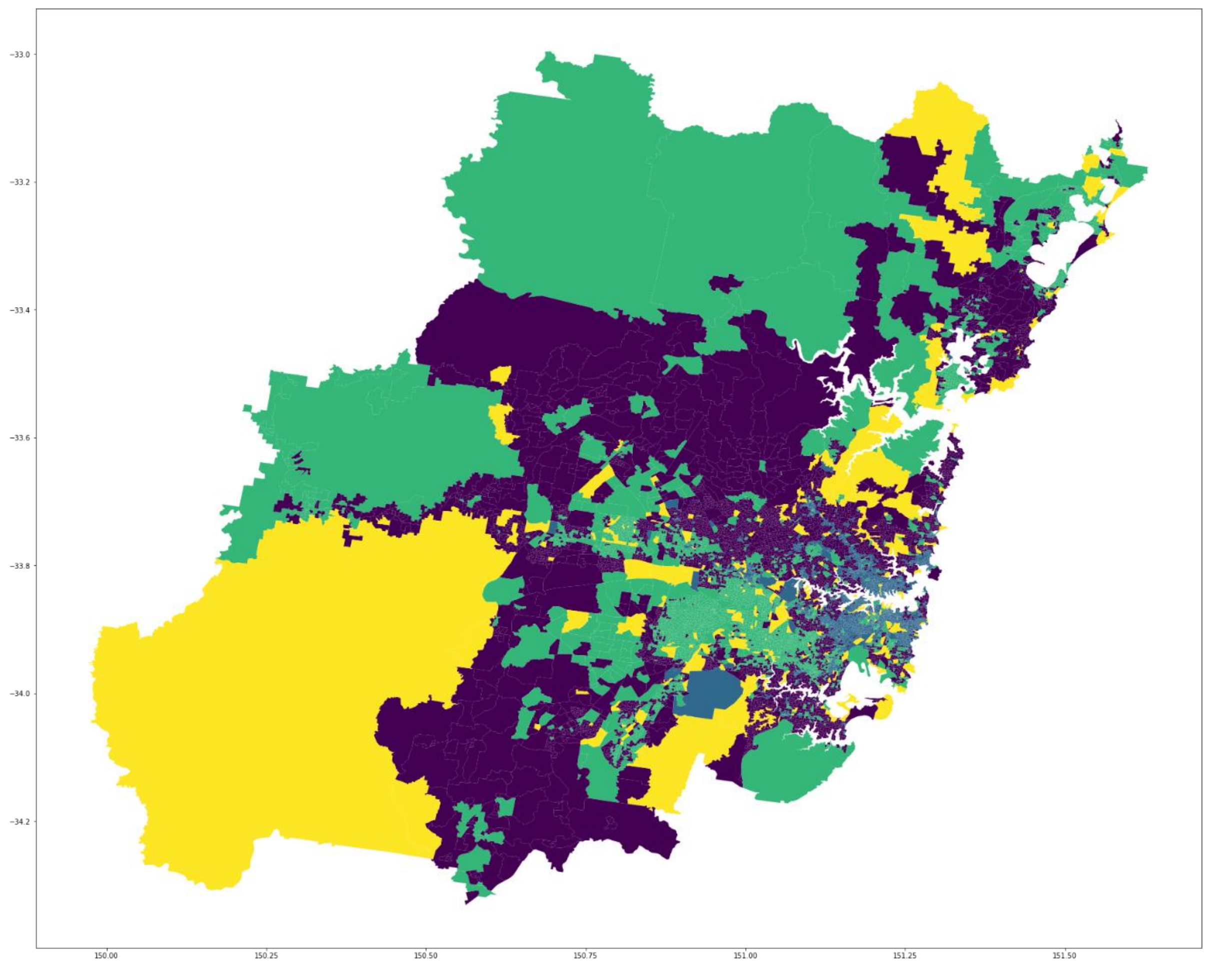
e.g. Cluster with label '1'



e.g. Cluster with label '3'

Plotting your clusters – on a map

```
%matplotlib inline
import geopandas as gpd

# Join the labels of k-means to previous dataset with SA1 IDs.

subset_df['labels'] = labels
subset_df

# Load in SA1 shapefile
lsoas_link = 'C:/Users/z5154264/OneDrive - UNSW/ProgrammableCities/2020/Final
Week/Geodemographics/SA1_2016_GREATER_SYDNEY.shp'
lsoas = gpd.read_file(lsoas_link)

# Fix ID to be same datatype between shapefile and original dataframe
lsoas['SA1_7DIG16'] = lsoas['SA1_7DIG16'].astype('int64')

# Merge on ID
joined_dfs = pd.merge(lsoas,subset_df,left_on='SA1_7DIG16',right_on='SA1_ID')

# Plot
f, ax = plt.subplots(1, figsize=(30, 30))
ax = joined_dfs.plot(axes=ax,column=labels)
plt.show()
```

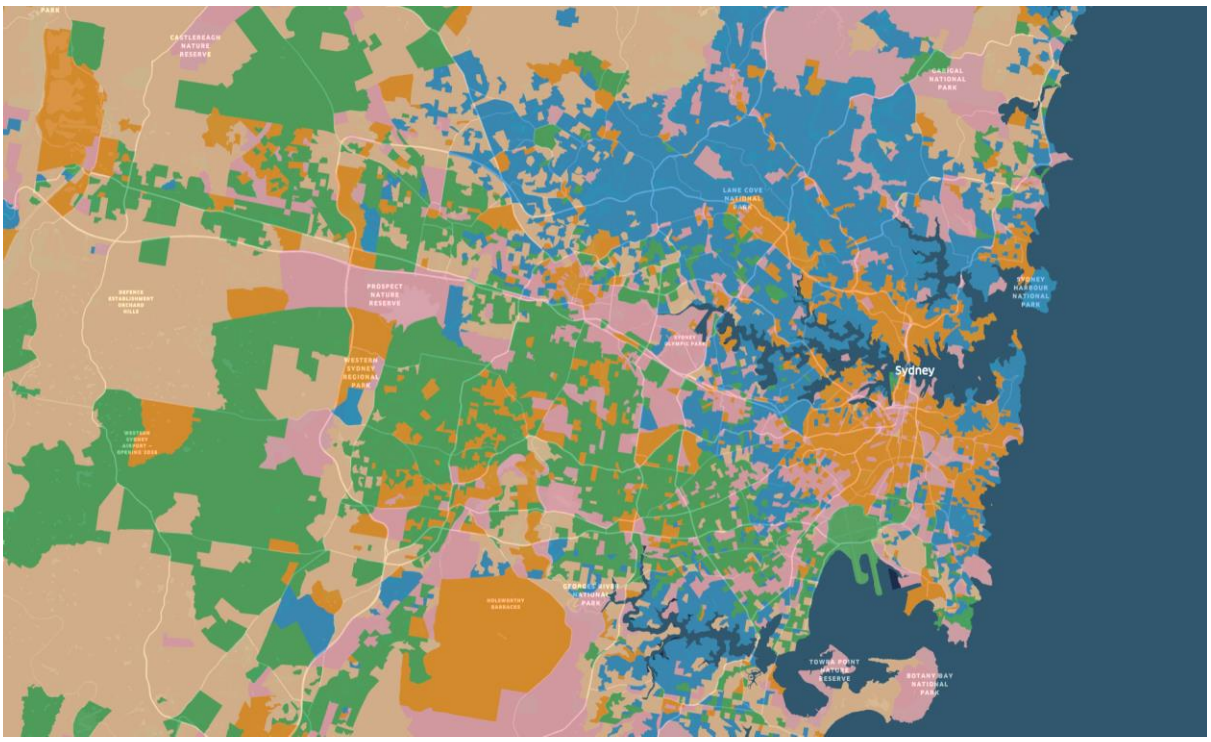


Export the geo-data

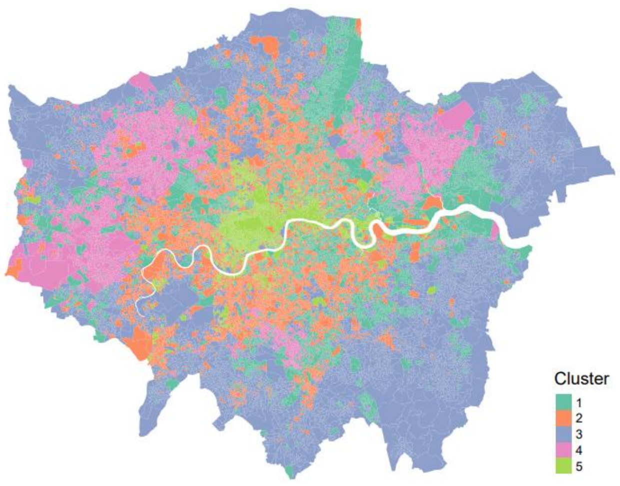
While you are refining your clusters, it may also be useful to style and filter them in a more dedicated GIS (ArcGIS / QGIS) or web visualisation platform (Kepler.gl). You can export through this script below.

```
joined_dfs.to_file("c:/data/sa1_clusters.geojson", driver='GeoJSON')
```

Here is an example of a set of clustered SA1s – can you think of names to give each colour?
Example interactive results: https://kepler.gl/demo/map?mapUrl=https://dl.dropboxusercontent.com/s/2eq9oksgertawsl/keplergl_jc83zbb.json


Castlereagh Nature Reserve
Lane Cove National Park
Prospect Nature Reserve
Western Sydney Regional Park
Sydney Harbour National Park
Sydney
Towra Point Nature Reserve
Botany Bay National Park

Here is a similar exercise, but for London.



Naming the clusters

Formulating names for geodemographic groups is not easy and can be contentious. Essentially the names are derivative of the researcher's perceptions of the cluster centres and sometimes also consider their geographic distributions. Names should be informative, short and memorable. They should also not make unnecessary assumptions or be derogatory in any way. Look at all of your plots - can you derive names for your clusters?

It is not uncommon for one of your cluster groups to represent the typical traits of the entire dataset. This group would, therefore, have very moderate Z-scores.

Once you are happy with your clusters making sense, being nameable and looking sensible across all of your plots you are finished.

Conclusion

From this exercise, you should have produced a geodemographic classification with a set number of distinctive
groups. In almost every case, geodemographic groups should display some spatial pattern within urban areas,
this is because of the tendency for social groups to cluster. This tutorial introduced a number of useful techniques, but not all of the possible methods that are used to create geodemographic classifications. At each of the steps you, the classification builder, had to make analytical methodological decisions. Each step should be very carefully thought through in order to produce the optimum classification to be used for a particular purpose. For instance, the inclusion of a single additional variable may influence which cluster a particular case is assigned to. Therefore, we recommend further reading on the theory and statistics behind techniques used in geodemographic clustering. Some suggestions have been provided below.

References

Arabie, P., Hubert, L.J. and De Soete, G. Eds. (1996). *Clustering and Classification.* Singapore, World Scientific

Debenham, J. (2002) *Understanding Geodemographic Classification: Creating The Building Blocks For An Extension*. Working Paper. School of Geography, University of Leeds.

Everitt, B.S. & T. Hothorn. (2014). *A Handbook of Statistical Analyses Using R (3rd ed.)* Boca Raton, Chapman & Hall

Everitt, B.S., Landau, S. and Leese, M. (2001). *Cluster Analysis 4th Ed*. London, Arnold

Gale, C.G., Singleton, A., Bates, A.G. and Longley, P.A., 2016. Creating the 2011 area classification for output areas (2011 OAC). *Journal of Spatial Information Science*, 12, pp.1-27.

Harris R, Sleight P, Webber R. (2005). *Geodemographics: Neighbourhood Targeting and GIS*. Chichester, UK:
John Wiley and Sons.

Kabacoff, R. (2015). *R in Action: Data Analysis and Graphics with R*. Manning Publications Co., Greenwich, USA

Lansley, G and Cheshire, J (2016). *An Introduction to Spatial Data Analysis and Visualisation in R*. CDRC Learning Resources. Online:
https://data.cdrc.ac.uk/tutorial/an-introduction-to-spatial-data-analysis-and-visualisation-in-r

Leventhal, B. (2016). *Geodemographics for Marketers: Using Location Analysis for Research and Marketing*. London, Kogan Page Publishers.

Longley, P.A. (2017). Geodemographic Profiling. *The International Encyclopedia of Geography* Wiley and the American Association of Geographers (AAG).

Longley, P.A., Goodchild, M., Maguire, D.J. and Rhind, D.W. (2010). *Geographic Information Systems and Science*. 4th Edition. John Wiley & Sons

Spielman, S. and A.D. Singleton. (2015). An Open Geodemographic Classification of US Census Tracts." *Annals of the Association of American Geographers.*

Usuelli, M. (2014). *R Machine Learning Essentials*. Birmingham, Packt

http://geogale.github.io/2011OAC/ - this webpage provides additional material and information to aid use of
the 2011 OAC, including the R code used to build it

https://www.opengeodemographics.com/ - for details on open geodemographic products made in the UK

https://www.geodemographics.org.uk/ - a comprehensive directory of hand-selected websites for people interested in the application of geodemographics and geo-spatial analysis