Population Science

Francisco Rowe, Carmen Cabrera-Arnau, Elisabetta Pietrostefani

2/12/23

Table of contents

Welcome

This is the website for "Population Science". This is a course designed and delivered by Dr. Francisco Rowe, Dr. Carmen Cabrera-Arnau and Dr. Elisabetta Pietrostefani from the Geographic Data Science Lab at the University of Liverpool, United Kingdom. You will learn applied tools and cutting-edge analytical approaches to use digital footprint data to explore and understand human population trends and patterns, including supervised and unsupervised machine learning approaches, network analysis and causal inference methods.

 A compilation of this web course is hosted as a GitHub repository that you can access:

- As a download of a `.zip` file that contains all the materials.
- As an html website.
- As a pdf document
- As a GitHub repository.

Contact

Francisco Rowe - f.rowe-gonzalez [at] liverpool.ac.uk Senior Lecturer in Quantitative Human Geography Office 507, Roxby Building, University of Liverpool, Liverpool, L69 7ZT, United Kingdom.

Carmen Cabrera-Arnau - c.cabrera-arnau [at] liverpool.ac.uk Lecturer in Geographic Data Science Office 1/016, Roxby Building, University of Liverpool, Liverpool, L69 7ZT, United Kingdom.

Elisabetta Pietrostefani - e.pietrostefani [at] liverpool.ac.uk Lecturer in Geographic Data Science Office 6xx, Roxby Building, University of Liverpool, Liverpool, L69 7ZT, United Kingdom.

1 Overview

The module provides students with an introduction to the use of data science and digital footprint data to analyse human population dynamics. Established approaches to study population dynamics rely on traditional data sources, such as censuses and surveys. Digital footprint data have emerged as a novel source of information providing an opportunity to understand key societal population issues at an unprecedented temporal and spatial granularity at scale (Rowe 2021). Yet, these data represent major methodological challenges to traditional demographic approaches (Rowe 2021). Machine learning, artificial intelligence and data science approaches are needed to overcome these methodological challenges.

1.1 Aims

This module aims to:

- provide an introduction to fundamental theories of population science;
- introduce students to novel data and approaches to understanding population dynamics and societal change; and,
- equip students with skills and experience to conduct population science using computational, data science approaches.

1.2 Learning Outcomes

By the end of the module, students should be able to:

- gain an appreciation of relevant demographic theory to help interpret patterns of population change;
- develop an understanding of the types of demographic and social science methods that are essential for interpreting and analysing digital footprint data in the context of population dynamics;
- develop the ability to apply different methods to understand population dynamics and societal change;

- gain an appreciation of how population science approaches can produce relevant evidence to inform policy debates;
- develop critical awareness of modern demographic analysis and ethical considerations in the use of digital footprint data.

1.3 Feedback

Formal assessment of two computational essays. Written assignment-specific feedback will be provided within three working weeks of the submission deadline. Comments will offer an understanding of the mark awarded and identify areas which can be considered for improvement in future assignments.

Verbal face-to-face feedback. Immediate face-to-face feedback will be provided during computer, discussion and clinic sessions in interaction with staff. This will take place in all live sessions during the semester.

Online forum. Asynchronous written feedback will be provided via an online forum. Students are encouraged to contribute by asking and answering questions relating to the module content. Staff will monitor the forum Monday to Friday 9am-5pm, but it will be open to students to make contributions at all times. Response time will vary depending on the complexity of the question and staff availability.

1.4 Computational Environment

To reproduce the code in the book, you need the following software packages:

- R-4.2.2
- RStudio 2022.12.0-353
- Quarto 1.2.280
- the list of libraries in the next section

To check your version of:

- R and libraries run `sessionInfo()`
- RStudio click `help` on the menu bar and then `About`
- Quarto check the `version` file in the quarto folder on your computer.

To install and update:

- R, download the appropriate version from The Comprehensive R Archive Network (CRAN)
- RStudio, download the appropriate version from Posit

- Quarto, download the appropriate version from the Quarto website

1.4.1 List of libraries

The list of libraries used in this book is provided below:

- `tidyverse`
- `viridis`
- `viridisLite`
- `ggthemes`
- `patchwork`
- `showtext`
- `RColorBrewer`
- `lubridate`
- `tmap`
- `sjPlot`
- `sf`
- `sp`
- `kableExtra`
- `ggcorrplot`
- `plotrix`
- `cluster`
- `factoextra`

You need to ensure you have installed the list of libraries used in this book, running the following code:

```
list.of.packages.cran <- c( “tidyverse”, “viridis”, “viridisLite”, “ggthemes”, “patchwork”, “showtext”, “RColorBrewer”, “lubridate”, “tmap”, “sjPlot”, “sf”, “sp”, “kableExtra”, “ggcorrplot”, “plotrix”, “cluster”, “factoextra”)

new.packages.cran <- list.of.packages.cran[!(list.of.packages.cran %in% installed.packages()[,“Package”])] if(length(new.packages.cran)) install.packages(new.packages.cran)

for(i in 1:length(list.of.packages.cran)) { library(list.of.packages.cran[i], character.only = T) }
```

1.5 Assessment

The final module mark is composed of the *two computational essays*. Together they are designed to cover the materials introduced in the entirety of content covered during the semester. A computational essay is an essay whose narrative is supported by code and computational

results that are included in the essay itself. Each teaching week, you will be required to address a set of questions relating to the module content covered in that week, and to use the material that you will produce for this purpose to build your computational essay.

Assignment 1 (50%) refers to the set of questions at the end of Chapter 2, Chapter 3, Chapter 4 and Chapter 5. You are required to use your responses to build your computational essay. Each chapter provides more specific guidance of the tasks and discussion that you are required to consider in your assignment.

Assignment 2 (50%) refers to the set of questions at the end of Chapter 6, Chapter 7, Chapter 8, Chapter 9 and Chapter 10. You are required to use your responses to build your computational essay. Each chapter provides more specific guidance of the tasks and discussion that you are required to consider in your assignment.

1.5.1 Format Requirements

Both assignments will have the same requirements:

- Maximum word count: 2,000 words, excluding figures and references.
- Up to three maps, plot or figures (a figure may include more than one map and/or plot and will only count as one but needs to be integrated in the figure)
- Up to two tables.

Assignments need to be prepared in "*Quarto Document*" format (i.e. qmd extension) and then converted into a self-contained HTML file that will then be submitted via Turnitin. The document should only display content that will be assessed. Intermediate steps do not need to be displayed. Messages resulting from loading packages, attaching data frames, or similar messages do not need to be included as output code. Useful resources to customise your R notebook can be found on Quarto's website.

Two Quarto Document templates will be available via the module Canvas site.

Submission is electronic only via Turnitin on Canvas.

1.5.1.1 Marking criteria

The Standard Environmental Sciences School marking criteria apply, with a stronger emphasis on evidencing the use of regression models, critical analysis of results and presentation standards. In addition to these general criteria, the code and outputs (i.e. tables, maps and plots) contained within the notebook submitted for assessment will be assessed according to the extent of documentation and evidence of expertise in changing and extending the code options illustrated in each chapter. Specifically, the following criteria will be applied:

- 0-15: no documentation and use of default options.

- 16-39: little documentation and use of default options.
- 40-49: some documentation, and use of default options.
- 50-59: extensive documentation, and edit of some of the options provided in the notebook (e.g. change north arrow location).
- 60-69: extensive well organised and easy to read documentation, and evidence of understanding of options provided in the code (e.g. tweaking existing options).
- 70-79: all above, plus clear evidence of code design skills (e.g. customising graphics, combining plots (or tables) into a single output, adding clear axis labels and variable names on graphic outputs, etc.).
- 80-100: all as above, plus code containing novel contributions that extend/improve the functionality the code was provided with (e.g. comparative model assessments, novel methods to perform the task, etc.).

2 Introducing Population Science

2.1 Introduction

Population science sits at the intersection between population studies and data science. As the general field of population studies, population science seeks to quantitatively understand human populations, including the three key demographic processes of population change, namely fertility, mortality and migration. It seeks to understand the size, structure, temporal changes and spatial distribution of populations, and the drivers and impacts that underpin their variations and regularities. It considers the ways in which structural social, economic, political and environmental factors shape population trends. What is unique about population science is that it seeks to leverage on the ongoing digital revolution characterised by technological advances in computer processing, digitalised information storage capacity and digital connectivity (Hilbert and López 2011).

The digital revolution ushered in the 1990s has unleashed a data revolution. Technological advances in computational power, storage and digital network platforms have enabled the emergence of "Big Data" or "digital footprint data". These technological developments have enabled the production, processing, analysis and storage of large volumes of digital data. Analysing 1986-2007 data, Hilbert and López (2011) estimated that the world has already passed the point at which more data were being collected than could be physically stored. They estimated that the global general-purpose computing capacity grew at an annual rate of 58 percent between 1986 and 2007, exceeding that of global storage capacity (23 percent). We can now digitally captured and generated data that previously could not easily be recorded and stored.

The unprecedented amount of information that we can now capture through digital technology offers unique opportunities to advance our understanding of *micro* human behaviour (e.g. individual-level decision making, preferences and choices) and *macro* population processes (e.g. structural population processes and trends). Digital footprint data offer a continuous flow of information to capture human population dynamics at unprecedentedly fine spatial and temporal resolution in real or near real-time comprising entire social systems. We can capture and study micro individual behaviours such as online time use, purchasing behaviour, visitation patterns and public opinion from data sources, such as mobile phones, social media and retail website platforms. These behaviours can also be aggregated to shed light into macro structural processes and trends, such as urban mobility, consumer demand, transport usage, population ageing and decline. Fundamentally digital footprint data thus have the potential

to become a key pillar informing and supporting decision making. They can inform business to increase sales revenue, football clubs to improve team performance, and governments to tackle major societal issues, such as the COVID-19 pandemic and global warming, influencing policy, practice and governance structures.

Yet, the use of digital footprint data also poses major conceptual, methodological and ethical challenges (Rowe 2021). It is these challenges that motivated this module. Digital footprint data are a by-product of an administrative process or service, and it is not purposely collected for research. Turning raw digital footprint data into actionable, usable information thus requires a unique combination of technical computational expertise and subject-specific knowledge. Traditionally university programmes have tended to focus on providing technical training, such as statistics or on specific knowledge subjects. But they are rarely found as a single coherent package. This module aims to fills this gap by offering training in the use of digital footprint data, and sophisticated methodological approaches (including machine learning, artificial intelligence, network science and statistical methods) to tackle important population issues, such as population segmenting, decline and mobility. Access to digital footprint data are highly variable; hence, we do not focus on this here. However, we encourage users of this book to read a report put together by the Joint Research Centre (2022) identifying and discussing key data sources focusing population processes.

The name of this module *Population Science* reflects the inclusive and interdisciplinary perspective we hope to capture. The data revolution has led to the emergence of a range of sub-disciplines, seeking to leverage on the use of digital footprint data to study human behaviour and population processes. These emerging sub-disciplines have tended to focus on discipline-specific issues such as digital demography (Kashyap et al. 2022), or particular methodological approaches, such as the use of networks principles in computational social sciences (Lazer et al. 2009). Population science seeks to integrate these perspectives and provide a fertile framework for critique, collaboration and co-creation across these emerging areas of scholarship in the study of human population. And, of course, take a spatial perspective adopting geographic data science approaches (Singleton and Arribas-Bel 2019).

Specifically, this chapter aims to discuss key opportunities and challenges of digital footprint data to analyse human population dynamics. We place a particular focus on the challenges relating to privacy, bias and privacy issues. The chapter starts by defining digital footprint data before discussing the key opportunities offered by these data and the challenges they pose.

2.2 Defining digital footprint data

We define digital footprint data as:

the data recorded by digital technology resulting from the interactions of people among themselves or with their social and physical environment, and they can take the form of images, video, text and numbers.

Data footprint data are distinctive features in their volume, velocity, variety, exhaustiveness, resolution, relational nature and flexibility (Kitchin 2014). They can take different forms. Traditional data used to be mostly numeric. Digital footprint data has facilitated the collection, storage and analysis of text (e.g. Twitter posts), image (e.g. Instagram photos) and video (e.g. CCTV footage) data.

Multiple digital systems contribute to the storage and generation of digital footprint data. Kitchin (2014) identified three broad systems directed, automated and volunteered systems. Directed systems comprise digital administrative systems operated by a human recording data on places or people e.g. immigration control, biometric scanning and health records. Automated systems involve digital systems which automatically and autonomously record and process data with little human intervention e.g. mobile phone applications, electronic smartcard ticketing, energy smart meter and traffic sensors. Volunteered systems involve digital spaces in which humans contribute data through interactions on social media platforms (e.g. Twitter and Facebook) or crowdsourcing (e.g. OpenStreetMap and Wikipedia).

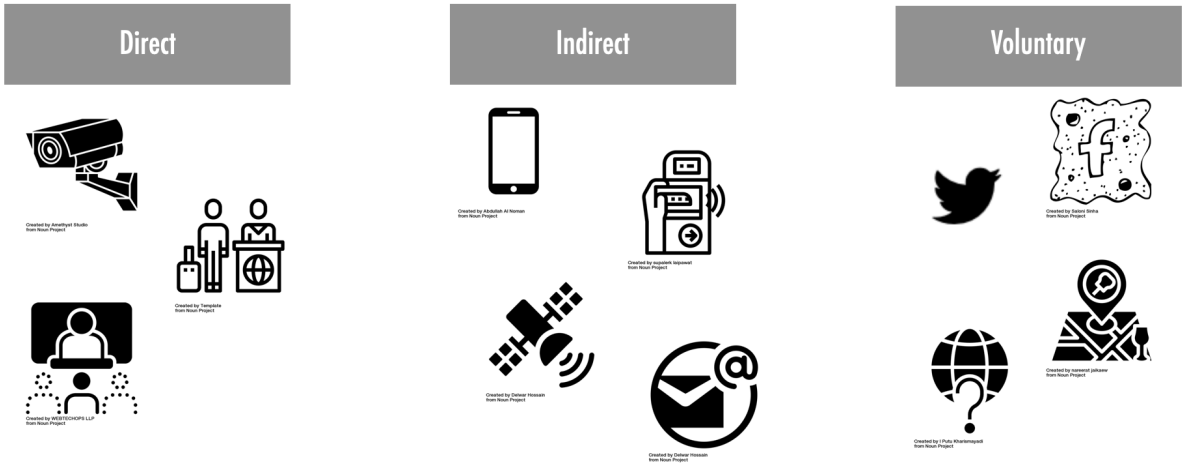


Figure 2.1: Digital footprint systems

2.3 Opportunities of digital footprint data

Digital footprint data offer unique opportunities for the analysis of human population patterns. As Rowe (2021) argues, digital footprint data offer three key promises in relation to traditional data sources, such as surveys and censuses. They generally provide greater spatio-temporal granularity, wider coverage and timeliness.

Digital footprint data offer high geographic and temporal *granularity*. Most digital footprint data are time-stamped and geographically referenced with high precision. Digital technology, such as mobile phone and geographical positioning systems enables the generation of a continuous steams of time-stamped location data. Such information thus provides an opportunity to trace and enhance our understanding human populations over highly granular spatial scales and time intervals, going beyond the static representation afforded by most traditional data sources. Spatial human interactions, and how people use and are influenced by their environment, can be analysed in a temporally dynamic way.

Digital footprint data provide extensive *coverage*. Contrasting to traditional random sampling, digital footprint data promise information on universal or near-universal population or geographical systems. Social media platforms, such as Twitter generate data to capture the entire universe of Twitter users. Satellite technology produces imagery snapshots to composite a representation of the Earth. Electronic smartcard ticketing systems produce information to capture the population of users in the system. Because the information is typically consistently collected and storage, the coverage of digital footprint data offer the potential to study human behaviour of entire systems at a global scale based on harmonised definitions, which is rarely possible using traditional data sources.

Digital footprint data are generated in *real-time*. Unlike traditional systems of data collection and release, digital footprint data can be streamed continuously in real- or near real-time. Commercial transactions are generally recorded on bank ledgers as bank card payments occur at retail shops. Individual mobile phone's location are captured as applications ping cellular antennas. Such information offer an opportunity to monitor and response to rapidly evolving situations, such as the COVID-19 pandemic (Green, Pollock, and Rowe 2021), natural disasters (Rowe 2022) and conflicts (Rowe, Neville, and González-Leonardo 2022).

We also loudly and clearly argue that while digital footprint data should be seen as a key asset to support government and business decision making processes, they should not be considered at the expenses of traditional data sources. Digital footprint data and traditional data sources should be used to complement each another. As indicated earlier, digital footprint data are the by-product of administrative processes or services. They were not designed with the aim of doing research. They require considerable work of data re-engineering to re-purpose them and turn them into an analysis-ready data product that can be used for further analysis (Arribas-Bel et al. 2021). Yet, as we will discuss below significant challenges remain. As the saying goes "*all data are dirty, but some data are useful*". This quote used in the data science community to convey the idea that data are often imperfect, but they can still be used to gain valuable insights. Our message is that digital footprint data and traditional data sources should be triangulated to leverage on their strengths and mitigate their weaknesses.

2.4 Challenges of digital footprint data

Digital footprint data also impose key conceptual, methodological and ethical challenges. In this section, we provide a brief explanation of challenges in these areas, focusing particularly on issues around biases, privacy, ethics and new methods. We focus on these issues because they are of practical importance and probably of most interest to the readers of this book. Excellent discussions have been written and, if you are interested in learning more about the challenges relating to digital footprint data, we recommend Kitchin (2014), Cesare et al. (2018), Lazer et al. (2020) and Rowe (2021).

2.4.1 Conceptual challenges

Conceptually, the emergence of digital footprint data has led to the rethinking and questioning of existing theoretical social science approaches (Franklin 2022). On the one hand, digital footprint data provide an opportunity to explore existing theories or hypotheses through different lens and test the consistency of existing beliefs. For example, economics theories discuss the existence of temporal and spatial equilibrium. Resulting hypotheses are generally tested through mathematical theoretical models or empirical analyses relying on temporally static data. The existence of equilibrium has thus remained hard to assess. Digital footprint data provide an opportunity to empirically test temporal and spatial equilibrium ideas based on suitable temporally dynamic data. They can enable the testing of cause and impact hypotheses, rather than only focusing on static associations.

On the other hand, digital footprint data sparked new questions. Digital footprint data provide data on previously unmeasured activities. Data now capture activities that were previously difficult to quantify, such as personal communications, social networks, search and information gathering, and location data. These data offer an opportunity to develop new questions expanding existing theories by looking inside the "black box" of households, organisations and markets. They may also open the door to developing entirely new questions such as the role of digital technology in shaping human behaviour, and the role of artificial intelligence on productivity and financial markets.

2.4.2 Methodological challenges

Methodologically, the need for a wide and new set of digital skills and expertise to handle, store and analyse large volumes of data is a key challenge. As indicated earlier, digital footprint data are not created for research purposes. They need to be reengineered for research. Large streams of digital footprint data cannot be stored on local memory. They can rarely be read as a single unit on a local computer and may involve performing the same task numerous times in regular basis, requiring therefore large storage, computational capacity and computer science expertise. The manipulation and storage of digital footprint data often require technical

expertise in data management systems, such as SQL, Google Cloud Storage and Amazon S3, as well as in efficient computing involving expertise in distributed computing systems and parallelisation frameworks. The analysis and modelling of digital footprint data may entail competencies in the application of machine learning and artificial intelligence. While these competencies generally form part of a computer science programme, they are rarely taught in an integrated framework focusing on addressing societal or business challenges relating to human populations - where the key focus is their application.

An additional methodological challenge is the presence of biases in digital footprint data. Digital footprint data are representative of a specific segment of the population but little is known which segments and how their representation varies across data sets and digital technology. Digital footprint data may comprise multiple sources of biases. They may reflect differences in the use of a digital device (e.g. mobile phone) and/or a piece of digital technology (e.g. a mobile phone application) Schlosser et al. (2021) . They may also reflect differences in *frequency* in the use of digital technology (e.g. number of times an individual uses a mobile phone application) - and this frequency may in turn reflect differences in algorithmic decisions embedded in digital platforms, such as suggesting content based on prior interactions to increase engagement with a given mobile phone application. Some work has been done on assessing biases as well as developing approaches to mitigate their influence Ribeiro, Benevenuto, and Zagheni (2020).

2.4.3 Ethical challenges

Privacy represents a major ethical challenge. Digital footprint data are highly sensitive, and hence, anonymisation and disclosure control are required. Individual records must be anonymised so they are not identifiable. The high degree of granularity and personal information of these records may and have been used in ethically questionable ways; for example, Cambridge Analytica used information of Facebook users to segment the population and target politically motivated content (Cadwalladr and Graham-Harrison 2018). Anonymising information, however, imposes a key challenge as there is a trade-off between accuracy and privacy (Petti and Flaxman 2020). Anonymisation may reduce the usability of data. The greater the degree of privacy, the lower is the degree of accuracy of the resulting data and *vice versa*. Identifying the optimal point balancing the privacy-accuracy trade-off is the key challenge. If doing incorrectly, we could end up drawing inferences that do not reflect the actual population processes displayed in the data, or have artificially been encoded in the data through noise or reshuffling. The application of data differential privacy to the US census provides a recent good example of this challenge. An emblematic case is New York's Liberty Island which has no resident population, but official US census reported 48 residents which was the result of adding statistical noise to the data, in order to enhance privacy.

2.5 Conclusion

Digital footprint data present unique oppotunites to enhance our understanding of population processes and support individual, business and government decisions to improve targeted processes and outcomes. Businesses have used digital footprint data to segment their consumer populations and improve their targeting of marketing content, products and ultimately increase sales and revenue (Dolega, Rowe, and Branagan 2021). Governments and health care institutions, particularly during the COVID-19 have leverage digital footprint data to monitor the spread of the pandemic and develop appropriate mitigation responses (Green, Pollock, and Rowe 2021). However, the use of digital footprint data poses major conceptual, methodological and ethical challenges - which need to be overcome to unleash their full potential. The aim of this book is to address of the key methodological challenges. In particular, this book seeks to provide applied training on the practical application of commonly used machine learning and artificial intelligence approaches to leverage on digital footprint data in the understanding of human behaviour and population processes.

3 Geodemographics

In this chapter we introduce the topic of geodemographics and geodemographic classifications. The chapter is based on the following references:

- Creating a Geodemographic Classification Using K-means Clustering in R (Guy Lansley and James Cheshire, 2018)
- Lecture 10 of the 2020-21 Work Book for the module GEOG0030 on Geocomputation, delivered at UCL Department of Geography.

Geodemographics is the statistical study of human populations based on their locations. It includes the application of geodemographic classifications (GDCs) or profiling whereby different locations are classified into groups based on the similarity in their demographic characteristics.

Assuming that the geodemographic characteristics of a group are an indicator of how the people in that group behave, GDC can be a very useful tool to predict the behavioral patterns of different regions. For this reason, geodemographics and GDC have applications in many areas, from marketing and retail to public health or service planning industries.

3.1 Dependencies

This chapter uses the libraries below. Ensure they are installed on your machine, then execute the following code chunk to load them:

```
#Support for simple features, a standardised way to encode spatial vector data
library(sf)
#Data manipulation
library(dplyr)
#A system for creating graphics
library(ggplot2)
#Easy viisualisation of a correlation matrix using ggplot2
library(ggcorrplot)
#Color maps designed to improve graph readability
library(viridis)
#Alternative way of plotting, useful for radial plots
```

```
library(plotrix)
#Methods for cluster analysis
library(cluster)
#Thematic maps can be generated with great flexibility
library(tmap)
#Provides some easy-to-use functions to extract and visualize the output of multivariate d
library(factoextra)
```

```
 #Obtain the working directory, where we will save the data directory
getwd()
```

```
[1] "/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202223/r4ps"
```

3.2 Data

3.2.1 Demographic data for Greater London Authority

In this Chapter we will be looking at data provided by London Datastore, a website created by the Greater London Authority (GLA) to distribute openly and freely London's data. In particular, we have prepared the file lsoa-data-clean.csv based on the LSOA Atlas, which contains demographic and related data for each Lower Layer Super Output Area (LSOA) in Greater London.

LSOAs are geographic hierarchies designed to improve the reporting of small area statistics in England and Wales. LSOAs are built from groups of contiguous Output Areas (OAs) and have been automatically generated to be as consistent as possible in population size, with a minimum population of 1,000. For this reason, their spatial extent varies depending on how densely populated a region is. The average population of an LSOA in London in 2010 was 1,722.

3.2.2 Import the data

In the code chunk below we load the dataset described above, lsoa-data-clean.csv as a data frame and call it *df_LSOA*. We will be generating some maps to show the geographical distribution of our data and results. To do this, we need the data that defines the geographical boundaries of the LSOAs. This data can be found in the form of a shapefile here. We have also stored this shapefile, called *LSOA_2011_Lodnon_gen_MHW.shp*, in the data folder of this workbook so you can import it directly as a data frame with simple features using the

st_read() function from the **sf** package. For more information on the sf package, check the documentation.

```
# Import LSOA demographic data for GLA
# The raw data can be obtained from link below, but it has been cleaned by Carmen Cabrera-
# https://data.london.gov.uk/dataset/lsoa-atlas
df_LSOA <- read.csv("./data/geodemographics/lsoa-data-clean.csv")

# Import LSOA boundaries for GLA
st_LSOA <- st_read("./data/geodemographics/LSOA_2011_London_gen_MHW/LSOA_2011_London_gen_M
```

```
Reading layer `LSOA_2011_London_gen_MHW' from data source
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202223/r4ps/data/geodemographi
  using driver `ESRI Shapefile'
Simple feature collection with 4835 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 503574.2 ymin: 155850.8 xmax: 561956.7 ymax: 200933.6
Projected CRS: OSGB36 / British National Grid
```

3.3 Preparing the data for GDC

3.3.1 Choice of geographic units

Normally, GDCs involve the analysis of aggregated demographic data into geographic units. Very small geographic units of data aggregation can provide more detailed results, but if the counts are too low, this could lead to re-identification issues.

As mentioned above, the data for this chapter is aggregated into LSOAs. The size of the LSOAs is small enough to produce detailed results and is also a convenient choice, since it is broadly used in the UK Census and other official data-reporting exercises.

We can visualise the LSOAs within GLA simply by plotting the geometry column of *sf_LSOA*, which can be selected with the function `st_geometry()`.

```
plot(st_geometry(st_LSOA), border=adjustcolor("gray20", alpha.f=0.4), lwd=0.6)
```

3.3.2 Variables of interest

Any classification task must be based on certain criteria that determines how elements are grouped into classes. For GDC, these criteria are demographic characteristics of the population located in the geographic units under study. In this case, we have prepared the file lsoa-data-clean.csv to contain some interesting demographic data corresponding to each LSOA. The data frame *df_LSOA* contains this data and we can visualise its first few lines by using the function `head()`:

```
head(df_LSOA[,1:4])
```

```
  Lower.Super.Output.Area    LSOA11NM MidYrPop MidYrPop0to15
1               E01000907 Camden 001A     1431         20.68
2               E01000908 Camden 001B     1564         18.16
3               E01000909 Camden 001C     1602         14.86
4               E01000912 Camden 001D     1589         16.17
5               E01000913 Camden 001E     1695         17.23
6               E01000893 Camden 002A     1563         17.08
```

As we can see, each row contains information about an LSOA and each column (starting from the third column) represents a demographic characteristic of the LSOA and the people living there. With the function `names()`, we can get the names of the columns in *df_LSOA*

```
names(df_LSOA[,1:8])
```

```
[1] "Lower.Super.Output.Area" "LSOA11NM"
[3] "MidYrPop"                "MidYrPop0to15"
[5] "MidYrPop16to29"          "MidYrPop30to44"
[7] "MidYrPop45to64"          "MidYrPop65"
```

The data frame *df_LSOA* contains many variables. As we can see above, they have summarised names. For a short description of what these names mean, we can load the file called Dictionary-lsoa-data-clean.csv:

```
df_dictionary <- read.csv("./data/geodemographics/Dictionary-lsoa-data-clean.csv")

head(df_dictionary)
```

```
          Label                                      Description
1       LSOA11NM                                    Name of LSOA
2       MidYrPop   Mid-year Population Estimates;All Ages;2011
3  MidYrPop0to15  Mid-year Population Estimates;Aged 0-15;2011
4 MidYrPop16to29 Mid-year Population Estimates;Aged 16-29;2011
5 MidYrPop30to44 Mid-year Population Estimates;Aged 30-44;2011
6 MidYrPop45to64 Mid-year Population Estimates;Aged 45-64;2011
```

For the purposes of this chapter, we will focus on just a few of these variables since this will make the results easier to interpret. In particular, we will look at variables related to ethnicity, country of birth, employment status and qualifications. Let us select the fields of interest:

```
df_LSOA <- df_LSOA[, c("LSOA11NM", "White", "MixedMulti", "Asian", "BlackAfricanCaribbean"
```

We can explore the summary statistics for each of the selected fields with the `summary()` function applied on the field of interest. For example, to obtain the summary statistics for the percentage of people belonging to the ethnic group 'Black/African/Caribbean/Black British', we can run the code below:

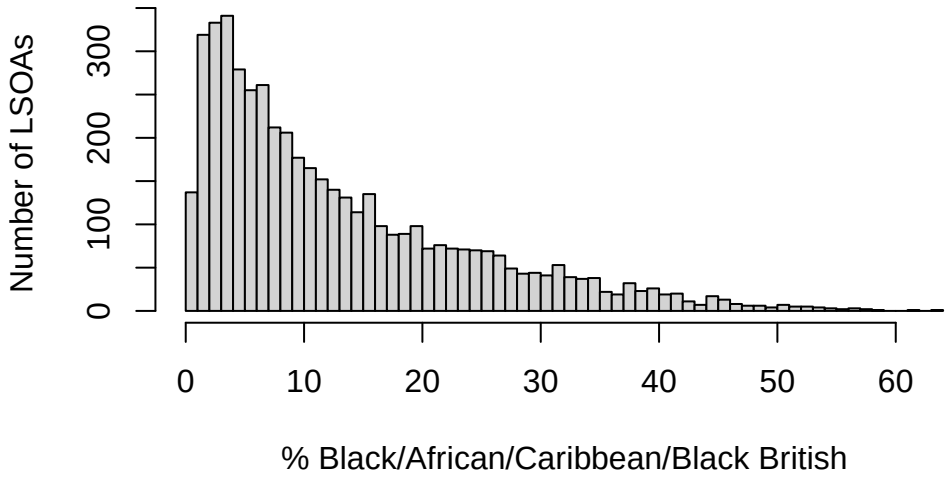
```
summary(df_LSOA$BlackAfricanCaribbean)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.10    4.30    9.50   13.05   18.90   63.70
```

This tells us that the mean or average percentage of people from this ethnic group in LSOAs within GLA is 13.05%. It also tells us that 63.70% of the population are Black/African/Caribbean/Black British in the LSOA with the maximum proportion of people belonging to this ethnic group.

To visualise the whole distribution of the variable 'Percentage of Black/African/Caribbean/Black British', we can plot a histogram:

```
hist(df_LSOA$BlackAfricanCaribbean, breaks=50, xlab="% Black/African/Caribbean/Black Briti
```



The histogram reveals that many LSOAs have a low proportion of Black/African/Caribbean/Black British people, but there are a few with more than 50% of their population belonging to this ethnic group.

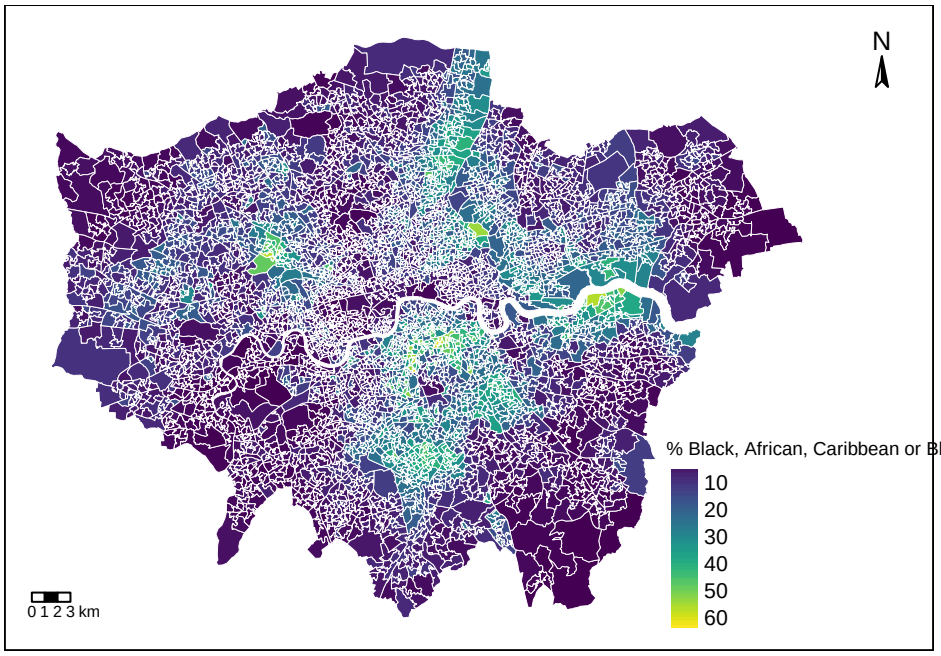
Now the question is whether the LSOAs with similar proportions of Black/African/Caribbean/Black British are also spatially close. To find out, we need to map the data. We can do this by joining the data frame *df_LSOA* with the data frame *st_LSOA* which contains the geographic boundaries of the LSOAs:

```
join_LSOA <- st_LSOA %>% left_join(df_LSOA, by='LSOA11NM')
```

If we plot the joined data frames using the `tmap` library functionalities, we can observe that, indeed there are specific regions within GLA with a high proportion of

Black/African/Caribbean/Black British people.

```
legend_title = expression("% Black, African, Caribbean or Black British")
map_ethnic = tm_shape(join_LSOA) +
  tm_fill(col = "BlackAfricanCaribbean", title = legend_title, text.size = 10, palette = v
  tm_layout(legend.position = c(0.71, 0.02), legend.title.size=0.7, inner.margins=c(0.05,
  tm_borders(col = "white", lwd = .01)  + # add borders
  tm_compass(type = "arrow", position = c("right", "top") , size = 1) + # add compass
  tm_scale_bar(breaks = c(0,1,2,3), text.size = 0.8, position =  c("left", "bottom")) # ad
map_ethnic
```



3.4 Standardisation

3.4.1 Across geographic units

Although LSOAs have been designed to have similar population sizes, the population figures fluctuate. And of course, if the population of a place is bigger or smaller, this can affect the figures corresponding to demographic characteristics (e.g. presumably, the larger the total population, the higher the number of people who are unemployed).

To counter the effect of variable population sizes across geographic units, we always need to

standardise the data so it is given as a proportion or a percentage. This has already done in the dataset lsoa-data-clean.csv, however, if you were to create your own dataset, you need to take this into account. To compute the right percentages, it is important to consider the right denominator. For example, if we are computing the percentage of people over the age of 65 in a given geographic unit, we can divide the number of people over 65 by the total population in that geographic unit, then multiply by 100. However, if we are computing the percentage of single-person households, we need to divide the number of single-person households by the total number of households (and not by the total population), then multiply by 100.

3.4.2 Variable standardisation

Data outliers are often present when analysing data from the real-world. These values are generally extreme and difficult to treat statistically. In GDC, they could end up dominating the classification process. To avoid this, we need to standardise the input variables as well, so that they all contribute equally to the classification process.

There are different methods for variable standardisation, but here we will achieve this by computing the Z-scores for each variable, i.e. for variable X, Z-score $= \dfrac{X - mean(X)}{std(X)}$ where $std()$ refers to standard deviation. In R, obtaining the Z-score of a variable is very simple with the function `scale()`. Since we want to obtain the Z-scores of all the variables under consideration, we can loop over the columns corresponding to the variables that we want to standardise:

```
# creates a new data frame
df_std <- df_LSOA
# extracts column names from df_std
colnames_df_std <- colnames(df_std)
# loops columns from position 1 : the last column
for(i in 2: ncol (df_std)){
df_std[, colnames_df_std[i]] <- scale(as.numeric(df_std[, colnames_df_std[i]]))
}
```

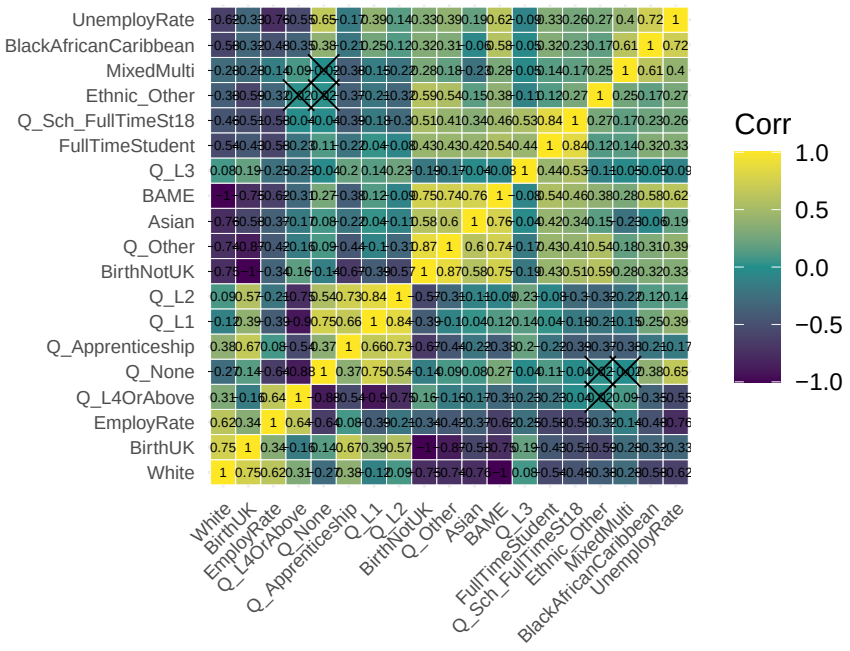
3.5 Checking for variable association

Before diving into the clustering process, it is necessary to check for variable associations. Two variables that are strongly associated could be conveying essentially the same information. Consequently, excessive weight could be attributed to the phenomenon they refer to in the clustering process. There are different techniques to check for variable association, but here we focus on the Pearson's correlation matrix.

Each row and column in a Pearson's correlation matrix represents a variable. Each entry in the matrix represents the level of correlation between the variables represented by the corresponding row and column. In R, a Pearson's correlation matrix can be created very easily with the `corr()` function, where the method parameter is set to "pearson". As a general rule, two variables with correlation coefficient greater than 0.8 or smaller than -0.8 are considered to be highly correlated. If this is the case, we might want to discard one of the two variables since the information they convey is redundant. However, in some cases, it might be reasonable to keep both variables if we can argue that they both have a similar but unique meaning.

The correlation coefficients by themselves are not enough to conclude whether two variables are correlated. Each correlation coefficient must be computed in combination with its p-value. For this reason, we also apply the `cor_pmat()` function below, which outputs a matrix of p-values corresponding to each correlation coefficient. Here, we set the confidence level at 0.95, therefore, p-values smaller than 0.05 are considered to be statistically significant. In the correlation matrix plot, we add crosses to indicate which correlation coefficients are not significant (i.e. those above 0.05). Those crosses indicate that there is not enough statistical evidence to reject the claim that the variables in the corresponding row and column are uncorrelated.

```
# Matrix of Pearson correlation coefficients
corr_mat <- cor(df_std[,c(colnames_df_std[2: ncol(df_std)])], method = "pearson")
# Matrix of p-values
corr_pmat <- cor_pmat(df_std[,c(colnames_df_std[2: ncol(df_std)])], method = "pearson", co
# Barring the no significant coefficient
ggcorrplot(corr_mat, tl.cex=7, hc.order = TRUE, outline.color = "white", p.mat = corr_pmat
```



Among the statistically significant values in the correlation matrix, we can see that BAME and White have a correlation of -1. So do BirthUK and BirthNotUK. We will therefore remove BAME and BrithUK from our dataset. We also see that Q_L4OrAbove has a strong negative correlation with Q_None and Q_L1. We will therefore remove two of these variables, for example Q_None and Q_L1.

```
#  Remove BAME, BirthUK, Q_None, Q_L1
data <- subset(df_std, select = -c(BAME, BirthUK, Q_None, Q_L1))
```

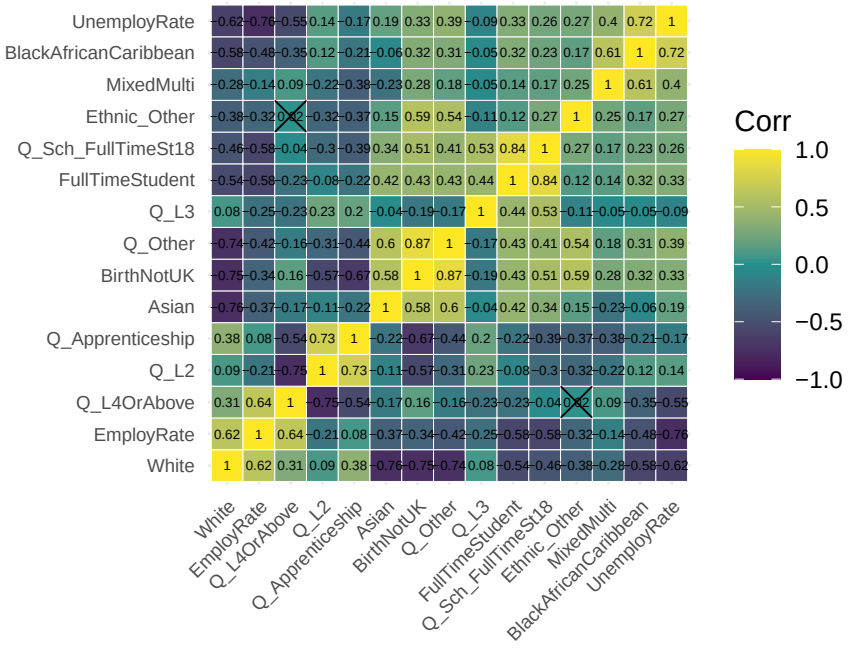
We can now perform a join of the resulting dataset with the variable st_LSOA, which stores the geographical units for the LSOAs. We perform this step so that we can later map the results

```
join_data <- st_LSOA %>% left_join(data, by='LSOA11NM')
```

And once again, we can check the Pearson correlation matrix of the resulting dataset. Obviously, now that we have removed some variables that were strongly correlated to others, the values of the correlation coefficients are not as high as before.

```
#Obtain column names from data
colnames_data <- colnames(data)
# Matrix of Pearson correlation coefficients
```

```
corr_mat_data <- cor(data[,c(colnames_data[2: ncol(data)])], method = "pearson")
# Matrix of p-values
corr_pmat_data <- cor_pmat(data[,c(colnames_data[2: ncol(data)])], method = "pearson")
# Barring the no significant coefficient
ggcorrplot(corr_mat_data, tl.cex=7, hc.order = TRUE, outline.color = "white", p.mat = corr
```



3.6 The clustering process

3.6.1 K-means

K-means clustering is a way of grouping similar items together. To illustrate the method, imagine you have a bag filled with vegetables, and you want to separate them into smaller bags based on their color, size and flavour. K-means would do this for you by first randomly selecting a number k of vegetables (you provide k, e.g. k=4), and then grouping all the other vegetables based on which of the k vegetables selected initially they are closest to in color, size and flavour. This process is repeated a few times until the vegetables are grouped as best as possible. The end result is k smaller bags, each containing veg of similar color, size and flavour. This is similar to how k-means groups similar items in a data set into clusters.

More technically, k-means clustering is actually an algorithm of unsupervised learning (we will learn more about this in Chapter 10) that partitions a set of points into k clusters, where k

is a user-specified number. The algorithm iteratively assigns each point to the closest cluster, based on the mean of the points in the cluster, until no point can be moved to a different cluster to decrease the sum of squared distances between points and their assigned cluster mean. The result is a partitioning of the points into k clusters, where the points within a cluster are as similar as possible to each other, and as dissimilar as possible from points in other clusters.

In R, k-means can be easily applied by using the function `k-means()`, where some of the required arguments are: the dataset, the number of clusters (which is called centers), the number of random sets to choose (nstart) or the maximum number of iterations allowed. For example, for a 4-cluster classification, we would run the following line of code:

```
Km <- kmeans(data[,c(colnames_data[2: ncol(data)])], centers=4, nstart=20, iter.max = 1000
```

3.6.2 Number of clusters

As mentioned above, the number of clusters k is a parameter of the algorithm that has to be specified by the user. Ultimately, there is no right or wrong answer to the question 'what is the optimum number of clusters?'. Deciding the value of k in the k-means algorithm can be a somewhat subjective process where in most cases, common sense is the most useful approach. For example, you can ask yourself if the obtained groups are meaningful and easy to interpret or if, on the other hand, there are too few or too many clusters, making the results unclear.

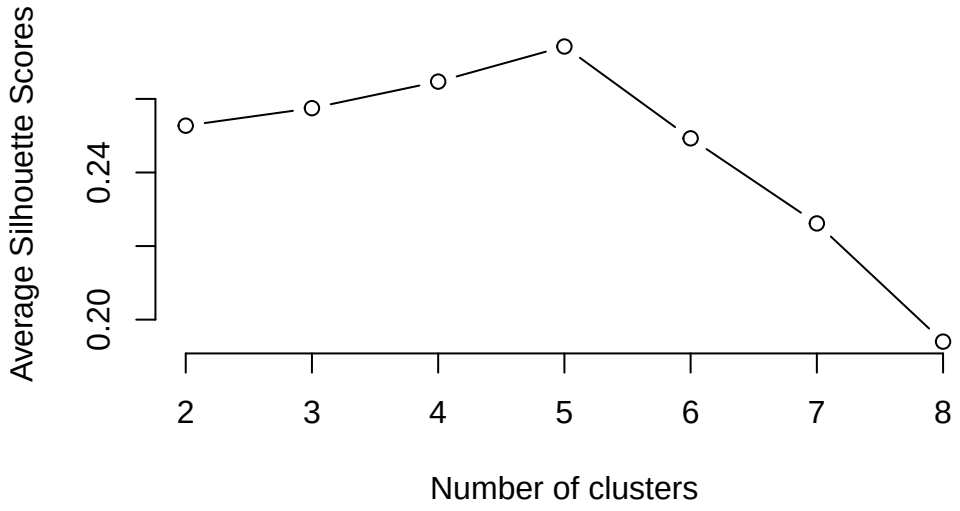
However, there are some techniques and guidelines to help us decide what the right number of clusters is. Here we explore the silhouette score method.

The **silhouette score** of a data point (in this case an LSOA and its demographic data) is a measure of how similar this data point is to the data points in its own cluster compared to the data points in other clusters. The silhouette score ranges from -1 to 1, with a higher value indicating that the data point is well matched to its own cluster and poorly matched to neighbouring clusters. A score close to 1 means that the data point is distinctly separate from other clusters, whereas a score close to -1 means the data point may have been assigned to the wrong cluster. Given a number of clusters k obtained with k-means, we can compute the average silhouette score over all the data points. Then, we can plot the average silhouette score against k. The optimal value of k will be the one with the highest k score.

You can run the code below to compute the average silhouette score corresponding to different values of k ranging from 2 to 8. The optimum number of clusters is given by the value of k at which the average silhouette is maximised.

```
silhouette_score <- function(k){
  km <- kmeans(data[,c(colnames_data[2: ncol(data)])], centers = k, nstart=5, iter.max = 1
  ss <- silhouette(km$cluster, dist(data[,c(colnames_data[2: ncol(data)])]))
```

```
  mean(ss[, 3])
}
k <- 2:8
avg_sil <- sapply(k, silhouette_score)
plot(k, type='b', avg_sil, xlab='Number of clusters', ylab='Average Silhouette Scores', fr
```



From the figure, we can see that the optimum k is 5, so we will take 5 as the number of clusters. Note, this number might be different when you run the programme since the clustering algorithm involves some random steps.

```
Km <- kmeans(data[,c(colnames_data[2: ncol(data)])], centers=5, nstart=20, iter.max = 1000
```

3.6.3 Other clustering methods

There are several other clustering methods apart from k-means. Each method has its own advantages and disadvantages, and the choice of method will ultimately depend on the specific data and clustering problem. We will not explore these methods in detail, but below we include some of their names and a brief description. If you want to learn about them, you can refer to the book "Pattern Recognition and Machine Learning" by Christopher Bishop (Bishop 2006).

- Fuzzy C-means: a variation of k-means where a data point can belong to multiple clusters with different membership levels.
- Hierarchical clustering: this method forms a tree-based representation of the data, where each leaf node represents a single data point and the branches represent clusters. A popular version of this method is agglomerative hierarchical clustering, where individual data points start as their own clusters, and are merged together in a bottom-up fashion based on similarity.
- DBSCAN: a density-based clustering method that groups together nearby points and marks as outliers those points that are far away from any cluster.
- Gaussian Mixture Model (GMM): GMMs are probabilistic models that assume each cluster is generated from a Gaussian distribution. They can handle clusters of different shapes, sizes, and orientations.

3.7 GDC results

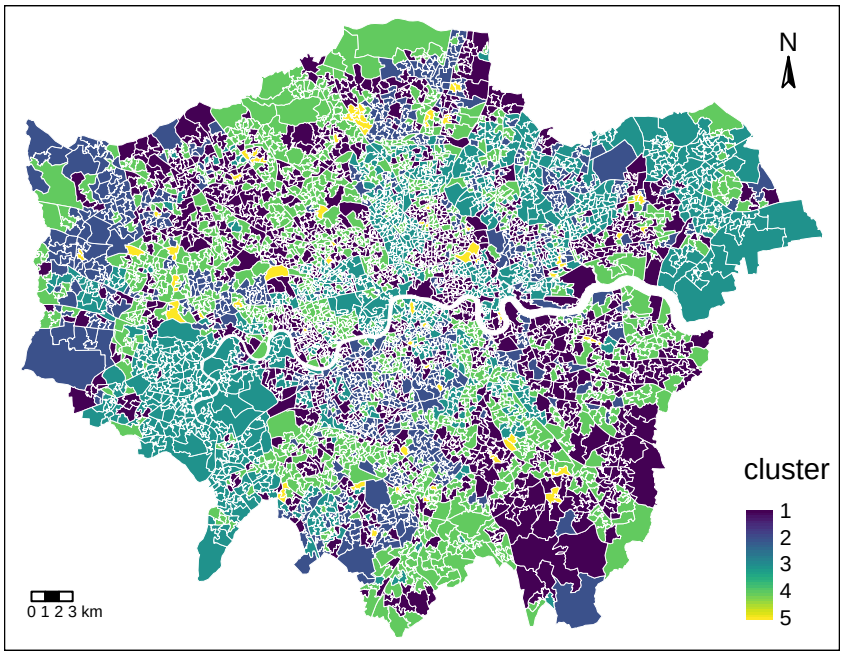
3.7.1 Mapping the clusters

Our LSOAs are now grouped into 5 clusters according to the similarity in their demographic characteristics. We can create a final data set based on *join_data* which includes the cluster where each geographical unit belongs to:

```
join_data_cluster <- join_data
join_data_cluster$cluster <- Km$cluster
```

Finally, we can plot the results of the clustering process on a map using functions from the tmap library:

```
map_cluster = tm_shape(join_data_cluster) +
  tm_fill(col = "cluster", title = "cluster", palette = viridis(256), style = "cont") + #
  tm_borders(col = "white", lwd = .01)  + # add borders
  tm_layout(legend.position = c(0.88, 0.02)) +
  tm_compass(type = "arrow", position = c("right", "top") , size = 1) + # add compass
  tm_scale_bar(breaks = c(0,1,2,3), text.size = 0.5, position =  c("left", "bottom")) # ad
map_cluster
```



Note: sometimes, the number of items in a cluster may be very small. In that case, you may want to merge to cluster to make the number of items in each group more homogeneous or perhaps change k in the k-means algorithm.

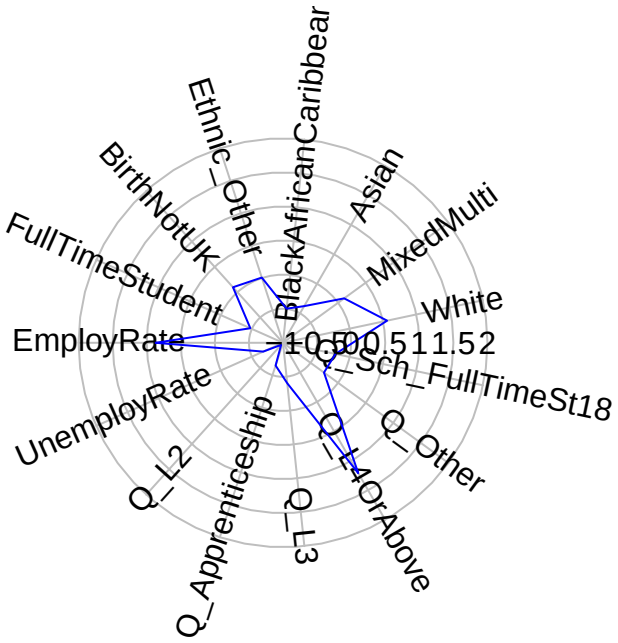
3.7.2 Cluster interpretation

The map above not only displays the clusters where each LSOA belongs, but it also shows that there is a tendency for LSOAs belonging to the same cluster to be geographically close. This indicates that people with similar demographic characteristics live close to each other. However, we still need to understand what each cluster represents.

The so-called cluster centers (kmCenters) are the data points that, within each cluster, provide a clear indication of the average characteristics the cluster where they belong based, of course, on the variables used in the classification. The data used in the clustering process was Z-score standardized, so the values of each variable corresponding to the cluster centers are still presented as Z-scores. Zero indicates the mean for each variable across all the data points in the sample, and values above or below zero indicate the number of standard deviations from the average. This makes it easy to understand the unique characteristics of each cluster relative to the entire sample. To visualise the characteristics and meaning of the clusters centers and their corresponding clusters, we use radial plots. Below we produce a radial plot for cluster 1. Can you see which variables are higher or lower than average in this cluster? If you want

to visualise the radial plot for other clusters, you will need to change the number inside the brackets of *KmCenters[1,]*.

```
# creates a radial plot for cluster 1
# the boxed.radial (False) prevents white boxes forming under labels
# radlab rotates the labels
KmCenters <- as.matrix(Km$centers)
KmCenters <- as.data.frame(KmCenters)
radial.plot(KmCenters[1,], labels = colnames(KmCenters),
boxed.radial = FALSE, show.grid = TRUE,
line.col = "blue", radlab = TRUE, rp.type="p", label.prop=0.9, mar=c(3,3,3,3))
```



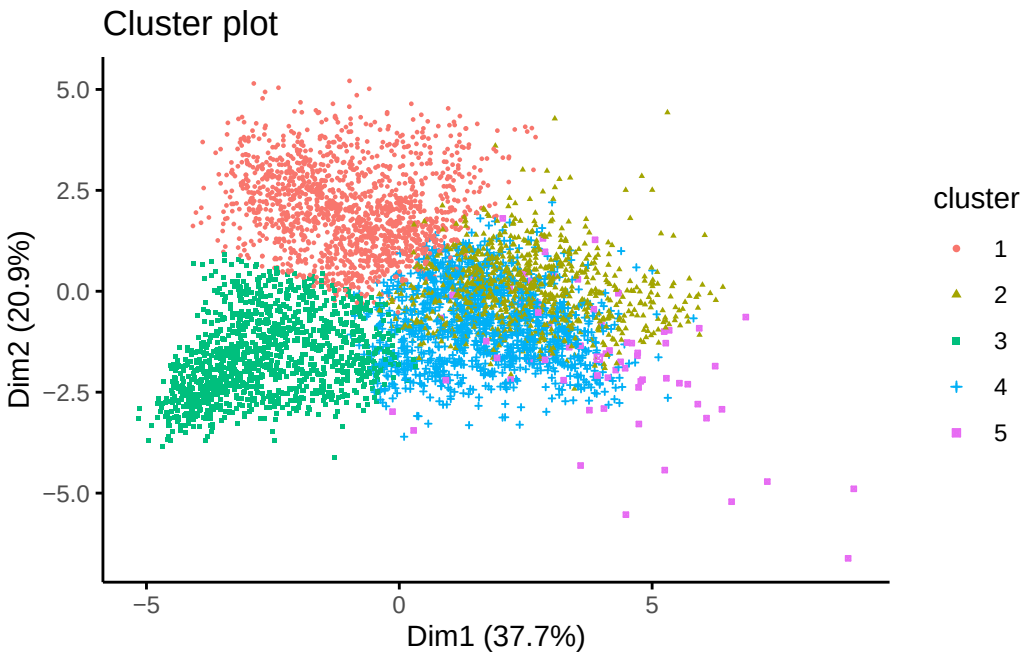
3.7.3 Testing

We will evalueate the fit of the k-means model with 5 clusters by creating an x-y plot of the of the first two principal components of each data point. Each point is coloured according to the cluster where it belongs. Remember that the aim of principal component analysis is to create the minimum number of new variables based on a combination of the original variables that can explain the variability in the data. The first principal component is the new variable that captures the most variability.

In the plot below, we can see the first and second principal components in the x and y axes

respectively, with the axis label indicating the amount of variability that these components are able to explain. To create the plot, we use the `fviz_cluster()` function from the factoextra library.

```
fviz_cluster(Km, data = data[,c(colnames_data[2: ncol(data)])], geom = "point", ellipse =
ggtheme = theme_classic())
```



There are obvious clusters in the plot, but some points are in the overlapping regions of two or more clusters, making it unclear to what cluster they should really belong. This does not mean that our classification is wrong, instead, it is a result of the fact that the plot is only representing two of the principal components, and there are other variables that are not captured in this 2-dimensional representation.

3.8 Questions

For this set of questions, we will be using the same datasets that we used for Chapter 3, the London LSOA dataset and the shapefile for the LSOA boundaries:

```
df_LSOA <- read.csv("./data/geodemographics/lsoa-data-clean.csv")
```

```r
# Import LSOA boundaries for GLA
st_LSOA <- st_read("./data/geodemographics/LSOA_2011_London_gen_MHW/LSOA_2011_London_gen_M
```

```
Reading layer `LSOA_2011_London_gen_MHW' from data source
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202223/r4ps/data/geodemographi
  using driver `ESRI Shapefile'
Simple feature collection with 4835 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 503574.2 ymin: 155850.8 xmax: 561956.7 ymax: 200933.6
Projected CRS: OSGB36 / British National Grid
```

This time, we will focus on demographic variables related to ethnicity, country of birth, housing ownership status, income and age group. Let us select these demographic variables for the questions below.

```r
df_LSOA <- df_LSOA[, c("LSOA11NM", "White", "MixedMulti", "Asian", "BlackAfricanCaribbean"
```

Prepare your data for a geodemographic classification (GDC). To do this, start by standardising the selected variables. Then, check for variable association using a correlation matrix. Discard any variables if necessary. Join the resulting dataset with the LSOA boundary data. Now you should be ready to group the data into clusters using the k-means algorithm. Based on the average silhouette score method, select the number of clusters for a GDC with k-means. Every time you apply the `kmeans()` function, you should set nstart=20 and iter.max=1000.

Essay questions:

1. Describe how you prepared your data for the GDC. There is no need to include figures, but you should briefly explain how you reached certain decisions. For example, did you discard any variables due to their strong association with other variables in the dataset? How did you pick the number of clusters for your GDC?
2. Map the resulting clusters and generate a radial plot for one of the clusters. You should create just one figure with as many subplots as needed.
3. Describe what you observe and comment on your results. Do you observe any interesting patterns? Do the results of this GDC agree with what you would expect? Justify your answer.

4 Sequence Analysis

In progress

5 Network Analysis

5.1 Dependencies

In this session we need some basic R packages before importing the data.

```
library(magrittr)
library(dplyr)

# An R package for network manipulation and analysis
library(igraph)
```

5.2 Data

5.2.1 The US Census dataset

Describe here the dataset used for this session. It has been cleaned beforehand by myself.

Each row corresponds to a origin-destination pair, the number or rows gives the total number of reported migratory movements.

5.2.2 Import the data

Before we start any analysis with the data, ensure to set the path to the directory where we are working. Please replace in the following line the path to the folder where you have placed the data file.

```
df <- read.csv("./data/networks/metro_to_metro_2015_2019_US_migration.csv")

#Ensure the MSA code is imported as a character and not as a number
df$MSA_Current_Code <- as.character(df$MSA_Current_Code)

#Include an additional column with the full name of the MSA in the format: Name, State
```

```r
df$MSA_Previous_Name_State <- paste0(df$MSA_Previous_Name, ', ', df$MSA_Previous_State)
df$MSA_Current_Name_State <- paste0(df$MSA_Current_Name, ', ', df$MSA_Current_State)

#Examine the first few rows of the dataset
head(df)
```

```
  MSA_Current_Code MSA_Current_Name MSA_Current_State
1            10180          Abilene                TX
2            10180          Abilene                TX
3            10180          Abilene                TX
4            10180          Abilene                TX
5            10180          Abilene                TX
6            10180          Abilene                TX
  MSA_Current_Population_1_Year_and_Over_Estimate
1                                         168,306
2                                         168,306
3                                         168,306
4                                         168,306
5                                         168,306
6                                         168,306
  MSA_Current_Population_1_Year_and_Over_MOE MSA_Previous_Code
1                                       300             10740
2                                       300             11100
3                                       300             12060
4                                       300             12220
5                                       300             12420
6                                       300             12580
                MSA_Previous_Name MSA_Previous_State
1                     Albuquerque                NM
2                        Amarillo                TX
3 Atlanta-Sandy Springs-Alpharetta                GA
4                  Auburn-Opelika                AL
5     Austin-Round Rock-Georgetown                TX
6        Baltimore-Columbia-Towson                MD
  MSA_Previous_Population_1_Year_and_Over_Estimate
1                                         902,213
2                                         262,574
3                                       5,753,503
4                                         153,728
5                                       2,045,336
6                                       2,766,530
  MSA_Previous_Population_1_Year_and_Over_MOE
```

```
1                                       2,916
2                                       1,866
3                                       9,607
4                                       1,654
5                                       5,648
6                                       6,154
  Movers_Metro_to_Metro_Flow_Estimate Movers_Metro_to_Metro_Flow_MOE
1                                  41                              47
2                                 244                              98
3                                 118                              95
4                                  30                              40
5                                 289                              85
6                                   8                              15
              MSA_Previous_Name_State MSA_Current_Name_State
1                      Albuquerque, NM           Abilene, TX
2                         Amarillo, TX           Abilene, TX
3 Atlanta-Sandy Springs-Alpharetta, GA           Abilene, TX
4                   Auburn-Opelika, AL           Abilene, TX
5     Austin-Round Rock-Georgetown, TX           Abilene, TX
6        Baltimore-Columbia-Towson, MD           Abilene, TX
```

We can obtain the total number of reported migratory movements with the following command:

```
nrow(df)
```

```
[1] 52930
```

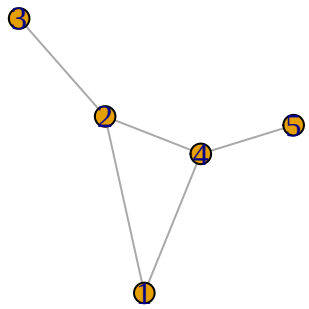
5.3 Creating networks

Before we start to analyse the data introduced in Section 5.2, let us first take a step back to consider the main object of study of this Chapter: the so-called networks. In the most general sense, a **network** (also known as a graph) is a structure formed by a set of objects which may have some connections between them. The objects are represented by **nodes** (a.k.a. vertices) and the connections between these objects are represented by **edges** (a.k.a. links). Networks are used as a tool to conceptualise many real-life contexts, such as the friendships between the members of a year group at school, the direct airline connections between cities in a continent or the presence of hyperlinks between a set of websites. In this session, we will use networks to model the migratory flows between US cities.

5.3.1 Starting from the basics

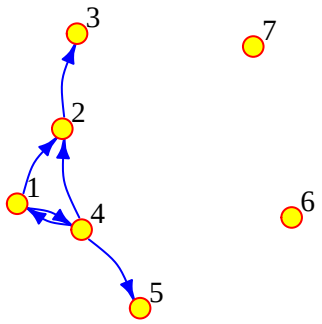
In order to create, manipulate and analyse networks in R, we will use the igraph package, which we imported in Section 5.2. We start by creating a very simple network with the code below. The network contains five nodes and five edges and it is undirected, so the edges do not have orientations. The nodes and edges could represent, respectively, a set of cities and the presence of migration flows between these cities in two consecutive years.

```
g1 <- graph( edges=c(1,2, 1,4, 2,3, 2,4, 4,5), n=5, directed=F ) # Creates an undirected n
# The number of nodes is given by argument n
# In this case, the node labels or IDs are represented by numbers 1 to 5
# The edges are specified as a list of pairs of nodes
plot(g1) # A simple plot of the network allows us to visualise it
```



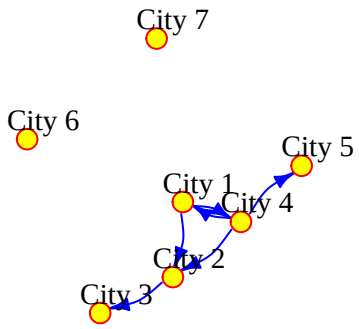
If the connections between the nodes of a network are non-reciprocal, the network is called directed. For example, this could correspond to a situation where there are people moving from city 1 to city 2, but nobody moving from city 2 to city 1. Note that in the code below we have not only added directions to the edges, but we have also added a few additional parameters to the plot function in order to customise the diagram.

```
g2 <- graph( edges=c(1,2, 1,4, 2,3, 4,1, 4,2, 4,5), n=7, directed=T ) # Creates a directed
#note that we now have edge 1,4 and edge 4,1 and that 2 of the nodes are isolated
plot(g2, vertex.frame.color="red",  vertex.label.color="black",
vertex.label.cex=0.9, vertex.label.dist=2.3, edge.curved=0.3, edge.arrow.size=.5, edge.col
```



The network can also be defined as a list containing pairs of named nodes. Then, it is not necessary to specify the number of nodes but the isolated nodes have to be included. The following code generates a network which is equivalent to the one above.

```
g3 <- graph( c("City 1","City 2", "City 2","City 3", "City 1","City 4",  "City 4","City 1"
plot(g3, vertex.frame.color="red",  vertex.label.color="black",
vertex.label.cex=0.9, vertex.label.dist=2.3, edge.curved=0.3, edge.arrow.size=.5, edge.col
```



5.3.2 Adding attributes

In R, we can add attributes to the nodes, edges and the network. To add attributes to the nodes, we first need to access them via the following command:

```
V(g3)
```

```
+ 7/7 vertices, named, from 0f01552:
[1] City 1 City 2 City 3 City 4 City 5 City 6 City 7
```

The node attribute *name* is automatically generated from the node labels that we manually assigned before.

```
V(g3)$name
```

```
[1] "City 1" "City 2" "City 3" "City 4" "City 5" "City 6" "City 7"
```

But other node attributes could be added. For example, the current population of the cities represented by the nodes:

```
V(g3)$population <- c(134000, 92000, 549000, 1786000, 74000, 8000, 21000)
```

Similarly, we can access the edges:

```
E(g3)
```

```
+ 6/6 edges from 0f01552 (vertex names):
[1] City 1->City 2 City 2->City 3 City 1->City 4 City 4->City 1 City 4->City 2
[6] City 4->City 5
```

and add edge attributes, such as the number of people moving from an origin to a destination city in two consecutive years. We call this attribute the *weight* of the edge, since if there is a lot of people going from one city to another, the connection between these cities has more importance or "weight" in the network.

```
{E(g3)$weight <- c(2000, 3000, 5000, 1000, 1000, 4000)}
```

We can examine the adjacency matrix of the network, which represents the presence of edges between different pairs of nodes. In this case, each row corresponds to an origin city and each column to a destination:

```
g3[] #The adjacency matrix of network g3
```

```
7 x 7 sparse Matrix of class "dgCMatrix"
       City 1 City 2 City 3 City 4 City 5 City 6 City 7
City 1      .   2000      .   5000      .      .      .
City 2      .      .   3000      .      .      .      .
City 3      .      .      .      .      .      .      .
City 4   1000   1000      .      .   4000      .      .
City 5      .      .      .      .      .      .      .
City 6      .      .      .      .      .      .      .
City 7      .      .      .      .      .      .      .
```

We can also look at the existing node and edge attributes.

```
vertex_attr(g3) #Node attributes of g3. Use edge_attr() to access the edge attributes
```

```
$name
[1] "City 1" "City 2" "City 3" "City 4" "City 5" "City 6" "City 7"
```

```
$population
[1]  134000   92000  549000 1786000   74000    8000   21000
```

Finally, it is possible to add network attributes

```
g3$title <- "Network of migration between cities"
```

5.4 Reading networks from data files

5.4.1 Preparing the data to create an igraph object

At the beginning of the chapter, we defined a data frame called *df* based on some imported data from the US Census about migratory movements between different US cities, or more precisely, between US Metropolitan Statistical Areas. This is a large data frame containing 52,930 rows, but how can we turn this data frame into a network similar to the ones that we generated in Section 5.3. The igraph function **graph_from_data_frame()** can do this for us. To find out more about this function, we can run the following command:

```
help("graph_from_data_frame")
```

As we can see, the input data for **graph_from_data_frame()** needs to be in a certain format which is different from our migration data frame. In particular, the function requires three arguments: 1) *d*, which is a data frame containing an edge list in the first two columns and any additional columns are considered as edge attributes; 2) *vertices*, which is either NULL or a data frame with vertex metadata (i.e. vertex attributes); and 3) *directed*, which is a boolean argument indicating whether the network is directed or not. Our next task is therefore to obtain 1) and 2) from the migration data frame called *df*.

Let us start with argument 1). Each row in *df* will correspond to an edge in the migration network since it contains information about a pair of origin and destination cities for two consecutive years. The names of the origin and destination cities are given by the columns in *df* called *MSA_Previous_Name* and *MSA_Current_Name*. In addition, the column called *Movers_Metro_to_Metro_Flow_Estimate* gives the number of people moving between the origin and the destination cities, so this will be the weight attribute of each edge in the migration network. Hence, we can define a data frame of edges which we will call *df_edges* that conforms with the format required by the argument 1) as follows:

```
#The pipe operator used below and denoted by %>% is a feature of the magrittr package, it
```

```r
# Creates the df_edges data frame with data from df and renames the columns as "origin", "
df_edges <- data.frame(df$MSA_Previous_Name_State, df$MSA_Current_Name_State, df$Movers_Me
  rename(origin = df.MSA_Previous_Name_State, destination = df.MSA_Current_Name_State, wei

#Ensure that the weight attribute is stored as a number and not as character
df_edges$weight <- as.numeric(gsub(",","",df_edges$weight))
```

For argument 2) we can define a data frame of nodes which we will call *df_nodes*, where each row will correspond to a unique node or city. To obtain all the unique cities from *df*, we can firstly obtain a data frame of unique origin cities, then a data frame of unique destinations, and finally, apply the **full_join()** function to these two data frames to obtain their union, which will be *df_nodes*. The name of the unique cities in *df_nodes* is in the column called *label*, the other columns can be seen as the nodes metadata.

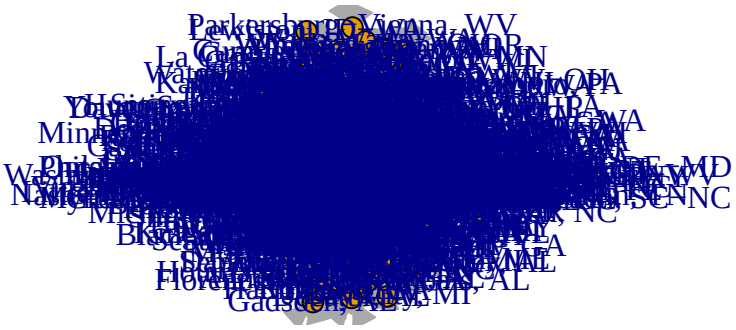
```r
df_unique_origins <- df %>%
  distinct(MSA_Previous_Name_State) %>%
  rename(name = MSA_Previous_Name_State)

df_unique_destinations <- df %>%
  distinct(MSA_Current_Name_State) %>%
  rename(name = MSA_Current_Name_State)

df_nodes <- full_join(df_unique_origins, df_unique_destinations, by = "name")
```

Finally, a directed migration network can be obtained with the following line of code. It should contain 386 nodes and 52,930 edges. You can test this yourself with the functions that you learnt in Section 5.3.

```r
g_US <- graph_from_data_frame(d = df_edges,
                                        vertices = df_nodes,
                                        directed = TRUE)
```

If we try to plot the network *g3* containing the migratory movements between all the US cities with the **plot()** function as we did before, we obtain a result which is rather undesirable…

```r
plot(g_US)
```



5.4.2 Filtering the data to create a subgraph

We will dedicate the entirety of next section to explore tools that can help us improve the visualisation of networks, since it is one of the most important aspects of network analysis. To facilitate the visualisation in the examples shown in Section 5.5, we will work with a subset of the full network called *g_US*. A way to create a subnetwork is to filter the original data frame. In particular, we will filter *df* to only include cities from a state, in this case, Minnesota. To filter, we use the **grepl()** function, which stands for grep logical. Both **grep()** and **grepl()** allow us to check whether a pattern is present in a character string or vector of a character string. While the **grep()** function returns a vector of indices of the element if a pattern exists in that vector, the **grepl()** function returns TRUE if the given pattern is present in the vector. Otherwise, it returns FALSE. In this case, we are filtering the dataset so that only the rows where the field MSA_Current_State is WA, which is the official abbreviation for Washington state.

```
df_sub <- df %>% filter(grepl('WA', MSA_Current_State)) %>% filter(grepl('WA', MSA_Previou
```

Then, we can prepare the data as we did before to create *gUS*. But, instead of basing the network on *df*, we will generate it from *df_sub*.

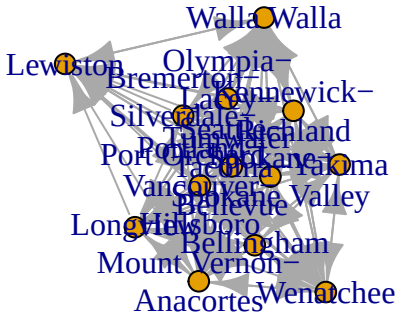
```
df_sub_edges <- data.frame(df_sub$MSA_Previous_Name, df_sub$MSA_Current_Name, df_sub$Mover
  rename(origin = df_sub.MSA_Previous_Name, destination = df_sub.MSA_Current_Name, weight

#Split long names into several lines for visualisation purposes
df_sub_edges$origin <- gsub("-", "-\n", df_sub_edges$origin)
df_sub_edges$destination <- gsub("-", "-\n", df_sub_edges$destination)

df_sub_edges$weight <- as.numeric(gsub(",","",df_sub_edges$weight))

df_sub_unique_origins <- df_sub %>%
  distinct(MSA_Previous_Name) %>%
  rename(name = MSA_Previous_Name)

df_sub_unique_destinations <- df_sub %>%
  distinct(MSA_Current_Name) %>%
  rename(name = MSA_Current_Name)

df_sub_nodes <- full_join(df_sub_unique_origins, df_sub_unique_destinations, by = "name")
df_sub_nodes$name <- gsub("-", "-\n", df_sub_nodes$name)

g_sub <- graph_from_data_frame(d = df_sub_edges,
                                        vertices = df_sub_nodes,
                                        directed = TRUE)
```

5.5 Network visualisation

5.5.1 Visualisation with igraph

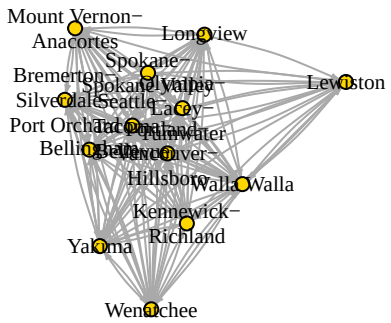
Let us start by generating the most basic visualisation of *g_sub*.

```
plot(g_sub)
```



This plot can be improved by changing adding a few additional arguments to the **plot()** function. For example, by just changing the color and size of the labels, the color and size of the nodes and the arrow size of the edges, we can already see some improvements.

```
plot(g_sub, vertex.size=10, edge.arrow.size=.2, edge.curved=0.1,
vertex.color="gold", vertex.frame.color="black",
vertex.label=V(g_sub)$name, vertex.label.color="black",
vertex.label.cex=.65)
```



But there are few more things we can do not only to improve the look of the diagram, but also to include more information about the network. For example, we can set the size of the nodes so that it reflects the total number of people that the corresponding cities receive. We can do this by adding a new node attribute, *inflow*, which is obtained as the sum of the rows of the adjacency matrix of *g_sub*.

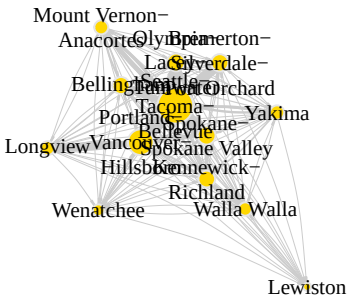
```
V(g_sub)$inflow <- rowSums(as.matrix(g_sub[]))
```

Below we set the node size based on the inflow attribute. Note the formula *0.4*(V(gsub)$inflow)^0.4*, where the power of 0.4 is chosen to scale the size of the nodes in such a way that the largest ones do not get excessively large and the smallest ones do not get excessively small. We also set the edge width based on its weight, which is the total number of people migrating from the origin and destination cities that it connects.

```
# Set node size based on inflow of migrants:
V(g_sub)$size <- 0.4*(V(g_sub)$inflow)^0.4
# Set edge width based on weight:
E(g_sub)$width <- E(g_sub)$weight/1200
```

Run the code below to discover how the aspect of the network has significantly improved with the modifications that we have introduced above.

```
plot(g_sub, vertex.size=V(g_sub)$size, edge.arrow.size=.15, edge.arrow.width=.2, edge.curv
vertex.color="gold", vertex.frame.color="gray90",
vertex.label=V(g_sub)$name, vertex.label.color="black",
vertex.label.cex=.65)
```



5.5.2 Visualisation of spatial networks

Firstly, we will import geographical data for the metropolitan and micropolitan statistical areas in the whole of the US, using the sf package. Here, we are only interested in the metropolitan areas so we will filter the data frame *cbsa_us* to keep only the metropolitan areas, i.e. those entries with value M1 for the column named *LSAD*.

```
library(sf)
```

```
Linking to GEOS 3.10.2, GDAL 3.4.2, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(stringr)

#Import core-based statistical areas https://www.census.gov/geographies/mapping-files/time
cbsa_us <- st_read("./data/networks/cb_2020_us_cbsa_500k/cb_2020_us_cbsa_500k.shp")
```

```
Reading layer `cb_2020_us_cbsa_500k' from data source
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202223/r4ps/data/networks/cb_
  using driver `ESRI Shapefile'
Simple feature collection with 939 features and 9 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -178.3347 ymin: 17.88328 xmax: -65.56427 ymax: 65.45352
Geodetic CRS:  NAD83
```

```
msa_us <- cbsa_us %>% filter(grepl('M1', LSAD)) #Filter the original data frame to obtain
```

We will now find the centroid of each MSA polygon and add columns to *msa_us* for the longitude and latitude of each centroid.

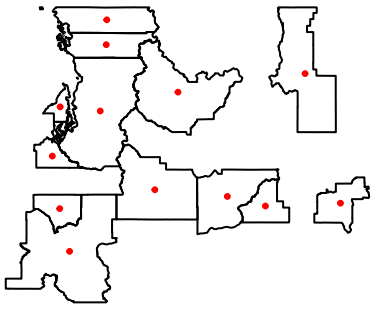
```
#Add longitude and latitude corresponding to centroid of each MSA polygon
msa_us$lon_centroid <- st_coordinates(st_centroid(msa_us$geometry))[,"X"]
msa_us$lat_centroid <- st_coordinates(st_centroid(msa_us$geometry))[,"Y"]
```

Since we are focusing on Washington state, let us filter *msa_us* so that it only includes data from Washington. This requires some data manipulation via the library stringr:

```
msa_us$NAME_ONLY <- gsub(",.*$", "", msa_us$NAME) #Create a new column with the name of th

#Long names of MSAs are split into lines for visualisation purposes
msa_us$NAME_ONLY <- gsub("-", "-\n", msa_us$NAME_ONLY)
msa_us$STATE <- substr(msa_us$NAME, nchar(msa_us$NAME)-1, nchar(msa_us$NAME)) #Create a ne

msa_sub <- msa_us %>% filter(grepl('WA', STATE)) #Filter to keep the metro areas belonging
```

We can now plot the polygons for the MSA belonging to Washington state as well as the centroids:

```
plot(st_geometry(msa_sub))
plot(st_centroid(msa_sub$geometry), add=TRUE, col="red", cex=0.5, pch=20)
```

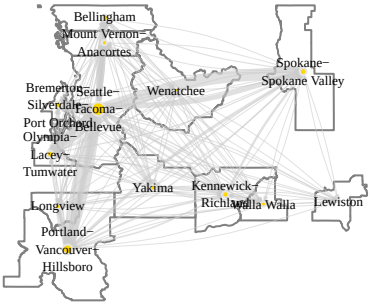


However, we still need to link this data to the network data that we obtained before. In order to incorporate the geographic information to the nodes of the migration subnetwork, we can join data from two data frames: *msa_sub*, which contains the geographic data, and *df_sub_nodes*, which contains the names of the nodes. To do this, we can use the function **left_join()** and then, select only the columns of interest. For more information on this magical function, check this link.

```
#Join the data frame of nodes df_sub_nodes with the geographic information of the centroid
df_sub_spatial_nodes <- df_sub_nodes %>% left_join(msa_sub, by = c("name" = "NAME_ONLY"))
```

```
lo <- as.matrix(df_sub_spatial_nodes[,2:3])
```

```
plot(st_geometry(msa_sub), border=adjustcolor("gray50"))
plot(g_sub, layout=lo, add = TRUE, rescale = FALSE, vertex.size=V(g_sub)$size, edge.arrow.
vertex.label=V(g_sub)$name, vertex.label.color="black",
vertex.label.cex=.45)
```



5.5.3 Alternative visualisations

In this session we have based our visualisations on igraph, however, there exist a variety of packages that would also allow us to generate nice plots of networks.

For example, COMPLETE

In addition, migration networks are particularly well-suited to be represented as a chord diagram. If you want to explore this type of visualisation, you can find further information on the official R documentation and also, for example, on this other link link.

5.6 Network metrics

Here we define some of the most important metrics that help us quantify different characteristics of a network. We will use the migration network for the whole of the US again, *g_US*. It has more nodes and edges than *g_sub* and consequently, its behaviour is richer and helps us illustrate better the concepts that we introduce in this section.

5.6.1 Density

The network **density** is defined as the proportion of existing edges out of all the possible edges. In a network with n nodes, the total number of possible edges is $n\times(n-1)$, i.e. the number of edges if each node was connected to all the other nodes. A density equal to 1 corresponds to a situation where $n \times (n-1)$ edges are present. A network with no edges at all would have density equal to 0. The line of code below tells us that the density of *g_sub* is approximately 0.33, meaning that about 33% of all the possible edges are present, or in other words, that there are migratory movements between almost a third of every pair of cities.

```
edge_density(g_US, loops=FALSE)
```

```
[1] 0.3283458
```

5.6.2 Reciprocity

The **reciprocity** in a directed network is the proportion of reciprocated connections between nodes (i.e. number of pairs of nodes with edges in both directions) from all the existing edges.

```
reciprocity(g_US)
```

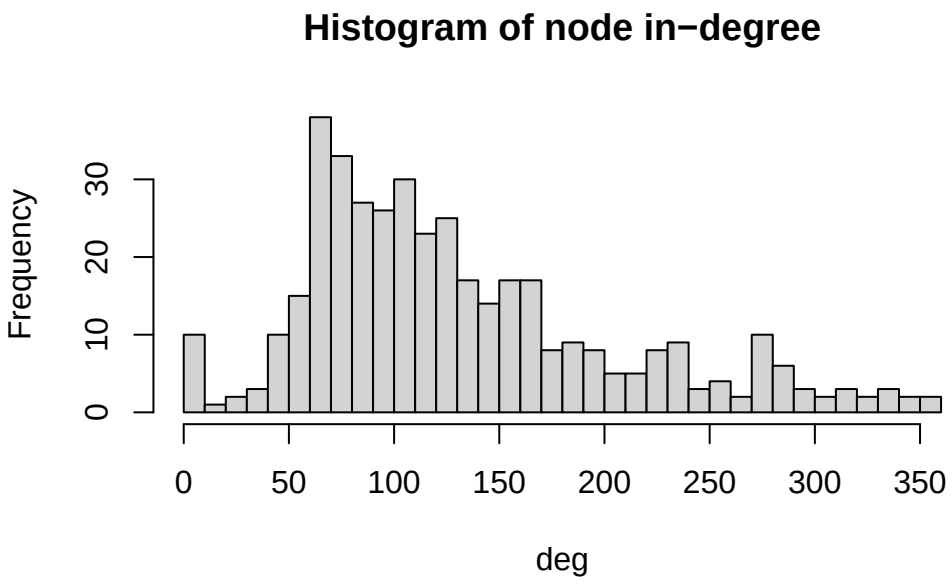
```
[1] 0.6067259
```

From this result, we conclude that about 62% of the pairs of nodes that are connected have edges in both directions.

5.6.3 Degree

The **total degree** of a node refers to the number of edges that emerge from or point at that node. The **in-degree** of a node in a directed network is the number of edges that point at it whereas the **out-degree** is the number of edges that emerge from it. The **degree()** functions, allows us to compute the degree of one or more nodes and allows us to specify if we are interested in the total degree, the in-degree or the out-degree.

```
# Compute degree of the nodes given by v belonging to graph g_US, in this case the in-degr
deg <- degree(g_US, v=V(g_US), mode="in")

#Produces histogram of the frequency of nodes with a certain in-degree
hist(deg, breaks = 30, main="Histogram of node in-degree")
```



As we can see in the histogram, many cities receive immigrants from 60-70 different cities. Very few cities receive immigrants from 300 or above cities. We can check which is the city with the maximum in-degree.

```
V(g_US)$name[degree(g_US, mode="in")==max(degree(g_US, mode="in"))]
```

```
[1] "Phoenix-Mesa-Chandler, AZ"
[2] "Washington-Arlington-Alexandria, DC-VA-MD-WV"
```

We actually obtain a tie between two: the MSA containing Phoenix in Arizona and the MSA containing Washington DC, which actually spans over four states. Their in-degree is 354 as we can see below.

```
degree(g_US, v=c("Phoenix-Mesa-Chandler, AZ"), mode="in")
```

```
Phoenix-Mesa-Chandler, AZ
                      354
```

```
degree(g_US, v=c("Washington-Arlington-Alexandria, DC-VA-MD-WV"), mode="in")
```

```
Washington-Arlington-Alexandria, DC-VA-MD-WV
                                          354
```

Note that the fact that these two cities have the largest in-degree does not necessarily mean that they are the ones receiving the largest number of migrants.

5.6.4 Distances

Shortest path between nodes, mean distance, diameter ,etc.

```
diameter(g_US)
```

```
[1] 36
```

5.6.5 Centrality

```
centr_degree(g_US, mode="in", normalized=T)
```

```
$res
  [1] 198 108 330 108 283 275 127 182 116 175 104  66 251 285 187 104 344 198
 [19] 259 123 258 231 141 197 345 331 280 174  66 113 237 159 110  78 148  96
 [37] 230  63 304 261 220 132  44  73  55  76 300 169 166  95 115 333 115 121
 [55] 125 205 297  79 123  66 144  78 291 330 148  67 176 243 237 150 209 240
 [73] 311 354 105  96 282 230 277 134 228 101 279 304 276 222 110 204 316  73
 [91] 113 110 149 336  73 122 245  81 101 128  45 128 354  78 164 138 157   0
[109]   0   0   0   0   0   0 172  33 163 276 148 132 271 167 104 103 105  26
[127] 247 128  36 232 286  53 112 200 160  25 113 261  47  20   0  75  70 170
[145]  61 194 202  74  81 113 170 164  72 105 221  65  92 255  92 271 216 192
[163]  55 180 196 181  97 178 170 122 157 117  83 151 232 122 288 228 133 163
[181] 187  70 227 186  71  68 282  64 127 185 222 240 144  54 276 156 274 157
[199] 207 235 219 127  69  88 128 158 103 156  74 116 190 102 100 163  90 237
[217] 104 319  93  67  68  77 179 146   0   0  76 136  68  56 131 166 115  69
[235] 164  88 106  93 217  99  71 140 200  70  82  81 103  94 146 168 155 109
[253]  94  95  62 113  98 103 152 123  62  98  87 165 187  76  66 117  69  93
[271]  85 108  92  90 123 116  88 116 133 213  50  69 110 151  61  99 129  67
[289]  72  74  65 151 161  63  83 154  93 129 117 107 132 115 145 150 155 125
[307] 134 130 178  42  62 139 130 151 185 161  54  86  99 100  52  70  74 108
[325]  75 119 140 128 115 121 134 139  43  52  78  47  68  69  90  87 143  80
[343]  66  83  50 140 159  67  80 106 114  85 117 107  80 123  95  87  93  72
```

```
[361]  86  81  81  89  63  83  93  54  66 124  55 134  60 119  75  77  81 107
[379]  77  56  78 113  51  37  74 102  80 145 101  62  87  83 100  64  59  68
[397]  99  67  41  76  51  46

$centralization
[1] 0.5544472

$theoretical_max
[1] 161202
```

5.7 Communities

5.8 Final visualisation

```
# V(g_US)$inflow <- rowSums(as.matrix(g_US[]))
# # Set node size based on inflow of migrants:
# V(g_US)$size <- 0.03*(V(g_US)$inflow)^0.1
# # Set edge width based on weight:
# E(g_US)$width <- E(g_US)$weight/1200
#
# #Join the data frame of nodes df_sub_nodes with the geographic information of the centro
# df_spatial_nodes <- df_nodes %>% left_join(msa_us, by = c("name" = "NAME")) %>% select(c
#
# lo <- as.matrix(df_spatial_nodes[,2:3])
#
# plot(st_geometry(msa_us), border=adjustcolor("gray50"))
# plot(g_US, layout=lo, add = TRUE, rescale = FALSE, vertex.size=V(g_US)$size, edge.arrow.
# vertex.label="", vertex.label.color="black",
# vertex.label.cex=.0)
```

6 Sentiment Analysis

In progress

7 Topic Modelling

In progress

8 Modelling Time

In progress

9 Assessing Interventions

In progress

10 Machine Learning

In progress

11 Data sets

In progress

References

Arribas-Bel, Dani, Mark Green, Francisco Rowe, and Alex Singleton. 2021. "Open Data Products-A Framework for Creating Valuable Analysis Ready Data." *Journal of Geographical Systems* 23 (4): 497–514. https://doi.org/10.1007/s10109-021-00363-5.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer.

Cadwalladr, Carole, and Emma Graham-Harrison. 2018. "Revealed: 50 Million Facebook Profiles Harvested for Cambridge Analytica in Major Data Breach." *The Guardian* 17 (1): 22.

Cesare, Nina, Hedwig Lee, Tyler McCormick, Emma Spiro, and Emilio Zagheni. 2018. "Promises and Pitfalls of Using Digital Traces for Demographic Research." *Demography* 55 (5): 1979–99. https://doi.org/10.1007/s13524-018-0715-2.

Dolega, Les, Francisco Rowe, and Emma Branagan. 2021. "Going Digital? The Impact of Social Media Marketing on Retail Website Traffic, Orders and Sales." *Journal of Retailing and Consumer Services* 60 (May): 102501. https://doi.org/10.1016/j.jretconser.2021.102501.

Franklin, Rachel. 2022. "Quantitative Methods II: Big Theory." *Progress in Human Geography* 47 (1): 178–86. https://doi.org/10.1177/03091325221137334.

Green, Mark, Frances Darlington Pollock, and Francisco Rowe. 2021. "New Forms of Data and New Forms of Opportunities to Monitor and Tackle a Pandemic." In, 423–29. Springer International Publishing. https://doi.org/10.1007/978-3-030-70179-6_56.

Hilbert, Martin, and Priscila López. 2011. "The World's Technological Capacity to Store, Communicate, and Compute Information." *Science* 332 (6025): 60–65. https://doi.org/10.1126/science.1200970.

Joint Research Centre. 2022. *Data innovation in demography, migration and human mobility*. LU: European Commission. Publications Office. https://doi.org/10.2760/027157.

Kashyap, Ridhi, R. Gordon Rinderknecht, Aliakbar Akbaritabar, Diego Alburez-Gutierrez, Sofia Gil-Clavel, André Grow, Jisu Kim, et al. 2022. "Digital and Computational Demography." http://dx.doi.org/10.31235/osf.io/7bvpt.

Kitchin, Rob. 2014. "Big Data, New Epistemologies and Paradigm Shifts." *Big Data & Society* 1 (1): 205395171452848. https://doi.org/10.1177/2053951714528481.

Lazer, David, Alex Pentland, Lada Adamic, Sinan Aral, Albert-László Barabási, Devon Brewer, Nicholas Christakis, et al. 2009. "Computational Social Science." *Science* 323 (5915): 721–23. https://doi.org/10.1126/science.1167742.

Lazer, David, Alex Pentland, Duncan J. Watts, Sinan Aral, Susan Athey, Noshir Contractor, Deen Freelon, et al. 2020. "Computational Social Science: Obstacles and Opportunities." *Science* 369 (6507): 1060–62. https://doi.org/10.1126/science.aaz8170.

Liang, Hai, and King-wa Fu. 2015. "Testing Propositions Derived from Twitter Studies: Generalization and Replication in Computational Social Science." Edited by Zi-Ke Zhang. *PLOS ONE* 10 (8): e0134270. https://doi.org/10.1371/journal.pone.0134270.

Petti, Samantha, and Abraham Flaxman. 2020. "Differential Privacy in the 2020 US Census: What Will It Do? Quantifying the Accuracy/Privacy Tradeoff." *Gates Open Research* 3 (April): 1722. https://doi.org/10.12688/gatesopenres.13089.2.

Ribeiro, Filipe N., Fabrício Benevenuto, and Emilio Zagheni. 2020. "How Biased Is the Population of Facebook Users? Comparing the Demographics of Facebook Users with Census Data to Generate Correction Factors." *12th ACM Conference on Web Science*, July. https://doi.org/10.1145/3394231.3397923.

Rowe, Francisco. 2021. "Big Data and Human Geography." http://dx.doi.org/10.31235/osf.io/phz3e.

———. 2022. "Using Digital Footprint Data to Monitor Human Mobility and Support Rapid Humanitarian Responses." *Regional Studies, Regional Science* 9 (1): 665–68. https://doi.org/10.1080/21681376.2022.2135458.

Rowe, Francisco, Ruth Neville, and Miguel González-Leonardo. 2022. "Sensing Population Displacement from Ukraine Using Facebook Data: Potential Impacts and Settlement Areas." http://dx.doi.org/10.31219/osf.io/7n6wm.

Schlosser, Frank, Vedran Sekara, Dirk Brockmann, and Manuel Garcia-Herranz. 2021. "Biases in Human Mobility Data Impact Epidemic Modeling." https://doi.org/10.48550/ARXIV.2112.12521.

Singleton, Alex, and Daniel Arribas-Bel. 2019. "Geographic Data Science." *Geographical Analysis* 53 (1): 61–75. https://doi.org/10.1111/gean.12194.

Zagheni, Emilio, and Ingmar Weber. 2015. "Demographic Research with Non-Representative Internet Data." Edited by Nikolaos Askitas and Professor Professor Klaus F. Zimmermann. *International Journal of Manpower* 36 (1): 13–25. https://doi.org/10.1108/ijm-12-2014-0261.