

# **Population Science**

Francisco Rowe, Carmen Cabrera-Arnau, Elisabetta Pietrostefani

3/12/23

# Table of contents

<b>Welcome</b>	<b>5</b>
Contact . . . . .	5
<b>1 Overview</b>	<b>6</b>
1.1 Aims . . . . .	6
1.2 Learning Outcomes . . . . .	6
1.3 Feedback . . . . .	7
1.4 Computational Environment . . . . .	7
1.4.1 List of libraries . . . . .	8
1.5 Assessment . . . . .	9
1.5.1 Format Requirements . . . . .	9
<b>2 Introducing Population Science</b>	<b>11</b>
2.1 Introduction . . . . .	11
2.2 Defining digital footprint data . . . . .	12
2.3 Opportunities of digital footprint data . . . . .	13
2.4 Challenges of digital footprint data . . . . .	15
2.4.1 Conceptual challenges . . . . .	15
2.4.2 Methodological challenges . . . . .	15
2.4.3 Ethical challenges . . . . .	16
2.5 Conclusion . . . . .	17
<b>3 Geodemographics</b>	<b>18</b>
3.1 Dependencies . . . . .	18
3.2 Data . . . . .	19
3.2.1 Demographic data for Greater London Authority . . . . .	19
3.2.2 Import the data . . . . .	19
3.3 Preparing the data for GDC . . . . .	20
3.3.1 Choice of geographic units . . . . .	20
3.3.2 Variables of interest . . . . .	21
3.4 Standardisation . . . . .	24
3.4.1 Across geographic units . . . . .	24
3.4.2 Variable standardisation . . . . .	25
3.5 Checking for variable association . . . . .	25

3.6	The clustering process . . . . .	28
3.6.1	K-means . . . . .	28
3.6.2	Number of clusters . . . . .	29
3.6.3	Other clustering methods . . . . .	30
3.7	GDC results . . . . .	31
3.7.1	Mapping the clusters . . . . .	31
3.7.2	Cluster interpretation . . . . .	32
3.7.3	Testing . . . . .	33
3.8	Questions . . . . .	34
<b>4</b>	<b>Sequence Analysis</b>	<b>36</b>
4.1	Dependencies . . . . .	36
4.2	Data . . . . .	37
4.2.1	Data wrangling . . . . .	43
4.2.2	Exploratory data analysis . . . . .	45
4.3	Application . . . . .	50
4.3.1	Defining outcome process . . . . .	51
4.3.2	Optimal matching . . . . .	52
4.3.3	Clustering . . . . .	60
4.3.4	Visualising . . . . .	61
4.4	Questions . . . . .	66
<b>5</b>	<b>Network Analysis</b>	<b>68</b>
5.1	Dependencies . . . . .	68
5.2	Data . . . . .	69
5.2.1	The US Census dataset . . . . .	69
5.2.2	Import the data . . . . .	69
5.3	Creating networks . . . . .	70
5.3.1	Starting from the basics . . . . .	70
5.3.2	Adding attributes . . . . .	73
5.4	Reading networks from data files . . . . .	75
5.4.1	Preparing the data to create an igraph object . . . . .	75
5.4.2	Filtering the data to create a subgraph . . . . .	77
5.5	Network visualisation . . . . .	79
5.5.1	Visualisation with igraph . . . . .	79
5.5.2	Visualisation of spatial networks . . . . .	81
5.5.3	Alternative visualisations . . . . .	84
5.6	Network metrics . . . . .	84
5.6.1	Density . . . . .	84
5.6.2	Reciprocity . . . . .	85
5.6.3	Degree . . . . .	85
5.6.4	Distances . . . . .	87
5.6.5	Centrality . . . . .	88

5.6.6	Hubs and authorities . . . . .	91
5.7	Questions . . . . .	93
<b>6</b>	<b>Sentiment Analysis</b>	<b>96</b>
6.1	Dependencies . . . . .	97
6.2	Data . . . . .	97
6.2.1	Text data structures . . . . .	98
6.2.2	Basic text data principles . . . . .	99
6.3	Sentiment Analysis . . . . .	108
6.3.1	Dictionary-based methods . . . . .	109
6.3.2	VADER . . . . .	115
6.4	Questions . . . . .	121
<b>7</b>	<b>Topic Modelling</b>	<b>123</b>
<b>8</b>	<b>Modelling Time</b>	<b>124</b>
<b>9</b>	<b>Assessing Interventions</b>	<b>125</b>
<b>10</b>	<b>Machine Learning</b>	<b>126</b>
<b>11</b>	<b>Data sets</b>	<b>127</b>
	<b>References</b>	<b>128</b>

# 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.

The website is *free to use* and is licensed under the [Attribution-NonCommercial-NoDerivatives 4.0 International](#). 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 2021b). Yet, these data represent major methodological challenges to traditional demographic approaches (Rowe 2021b). 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`
- `igraph`
- `stringr`

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", "patch-
work", "showtext", "RColorBrewer", "lubridate", "tmap", "sjPlot", "sf", "sp",
"kableExtra", "ggcorrplot", "plotrix", "cluster", "factoextra", "igraph", "stringr")

new.packages.cran <- list.of.packages.cran[!(list.of.packages.cran %in% in-
stalled.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], charac-
ter.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 capture and generate 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 unprecedently 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 2021b). 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).

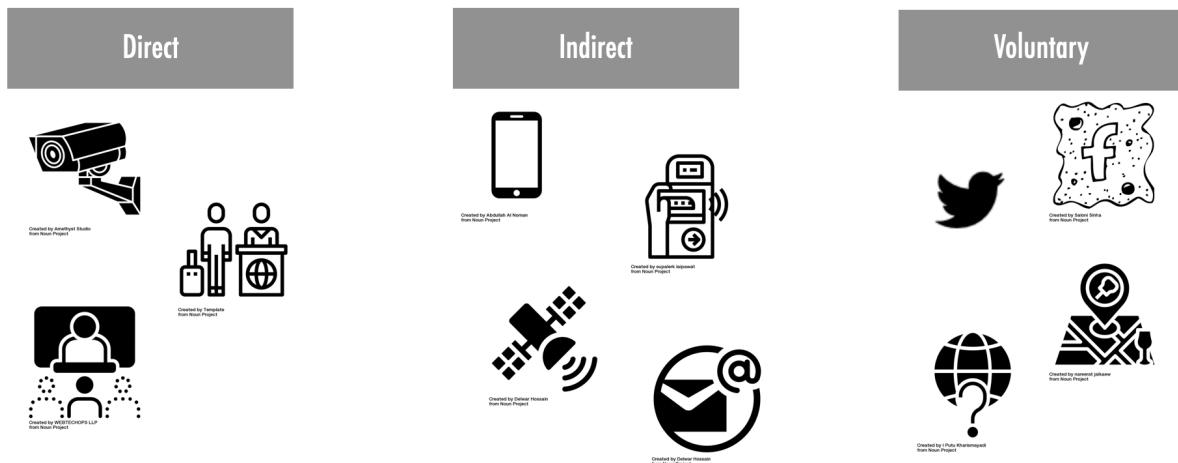


Figure 2.1: Digital footprint systems

## 2.3 Opportunities of digital footprint data

Digital footprint data offer unique opportunities for the analysis of human population patterns. As Rowe (2021b) 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 streams 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 2022b) 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 (2021b).

### **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 opportunities 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 visualisation 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\_London\_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_MH
```

```
Reading layer `LSOA_2011_London_gen_MHW' from data source
`/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202223/r4ps/data/geodemograph...
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)

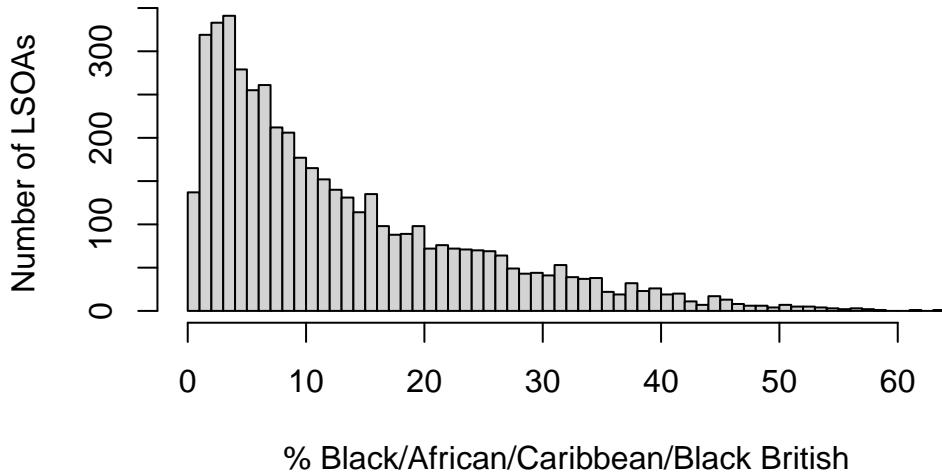
```

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.

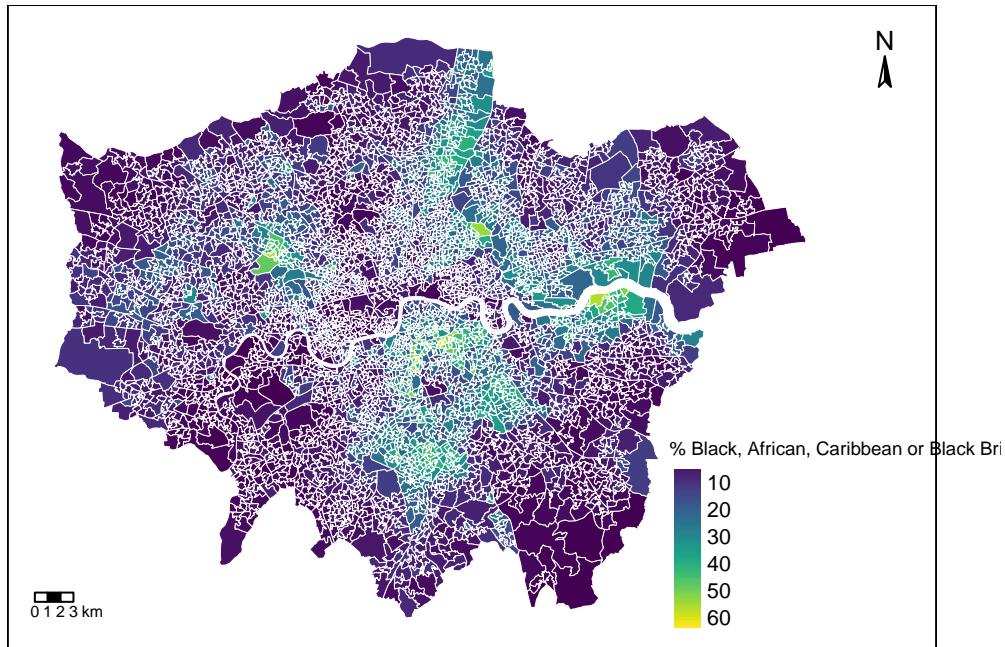
Now the question is whether the LSOAs with similar proportions of Black/African/Caribbean/Black British are also spatially close. To find out, we need to map the data. We can do this by joining the data frame *df\_LSOA* with the data frame *st\_LSOA* which contains the geographic boundaries of the LSOAs:

```
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 = viridis)
  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")) # add scale bar
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  $X$ ,  $Z\text{-score} = \frac{X - \text{mean}(X)}{\text{std}(X)}$  where  $\text{std}()$  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]]))
}
```

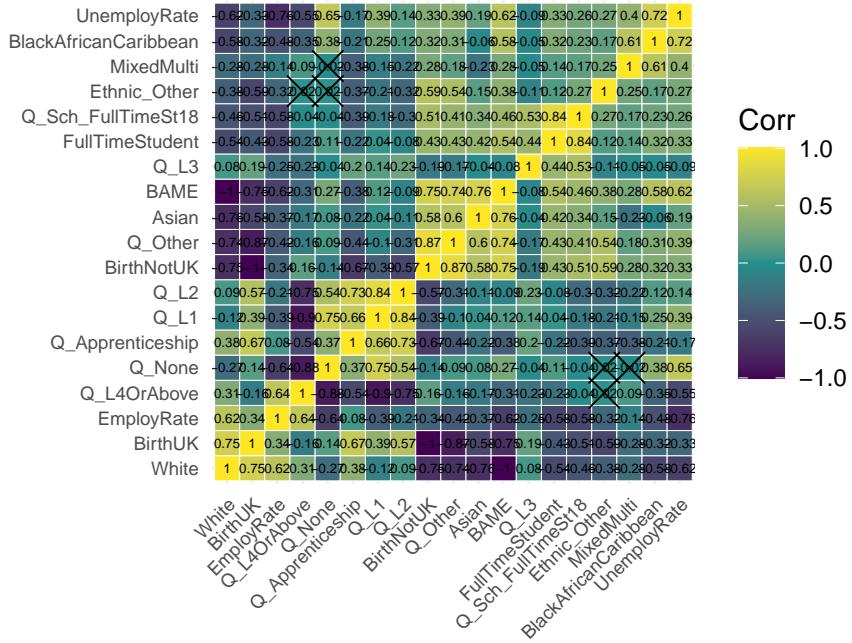
## 3.5 Checking for variable association

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 BirthUK 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')
```

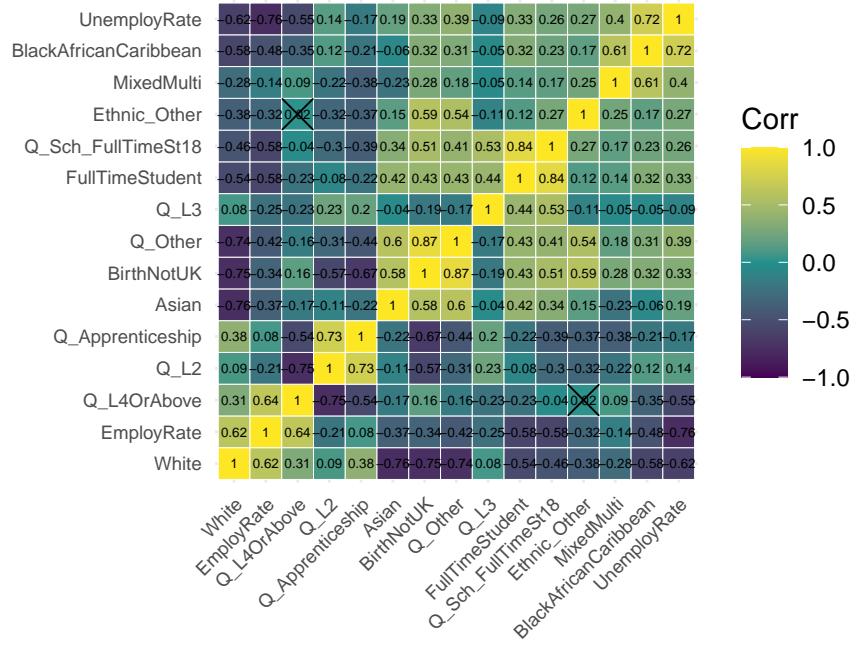
And once again, we can check the Pearson correlation matrix of the resulting dataset. Obviously, now that we have removed some variables that were strongly correlated to others, the values of the correlation coefficients are not as high as before.

```
#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.

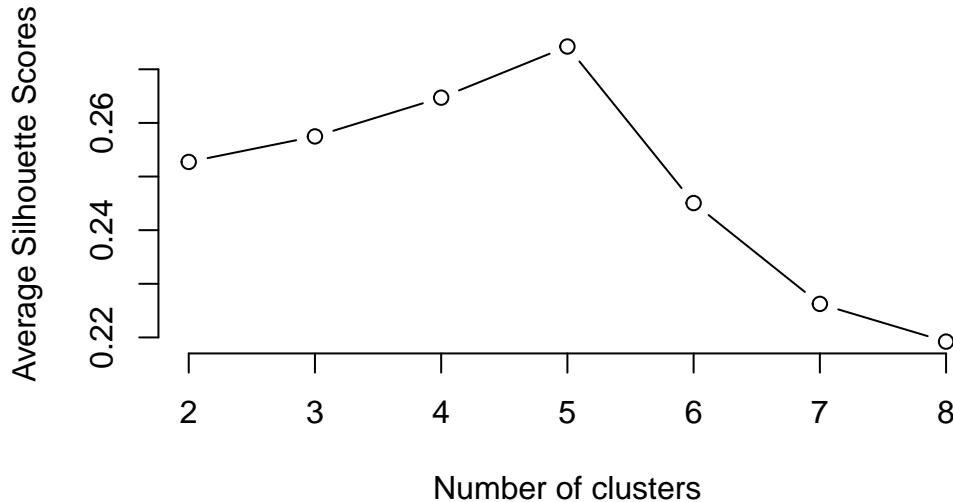
You can run the code below to compute the average silhouette score corresponding to different values of  $k$  ranging from 2 to 8. The optimum number of clusters is given by the value of  $k$  at which the average silhouette is maximised.

```
silhouette_score <- function(k){
  km <- kmeans(data[,c(colnames_data[2: ncol(data))]], centers = k, nstart=5, iter.max = 1000)
  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

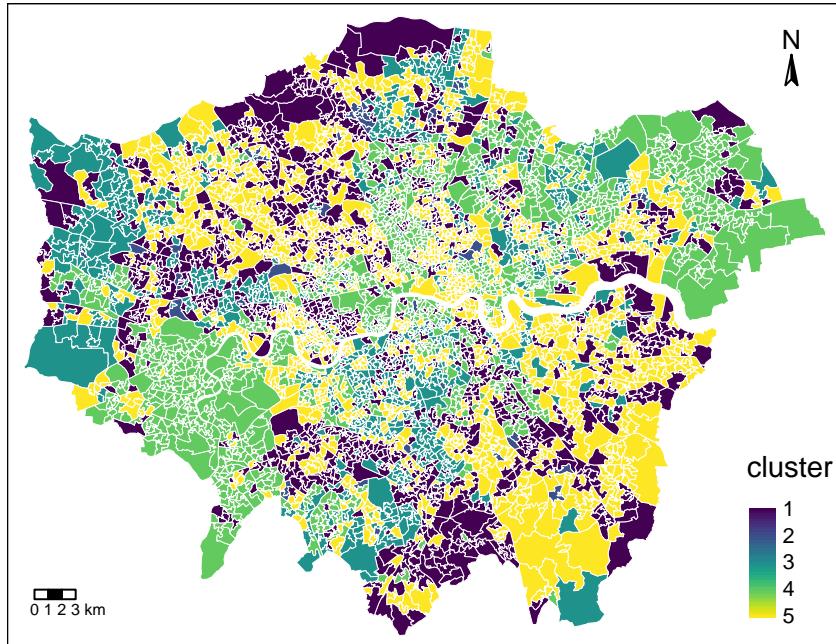
### 3.7.1 Mapping the clusters

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 two clusters to make the number of items in each group more homogeneous or perhaps change k in the k-means algorithm.

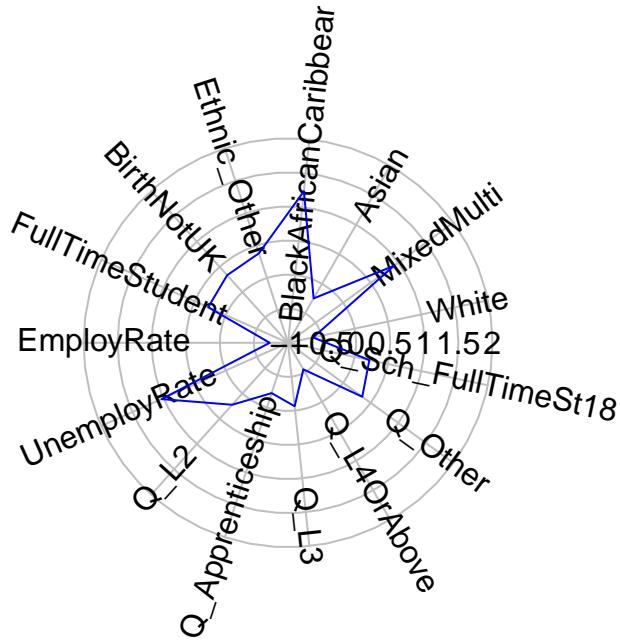
### 3.7.2 Cluster interpretation

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))
```



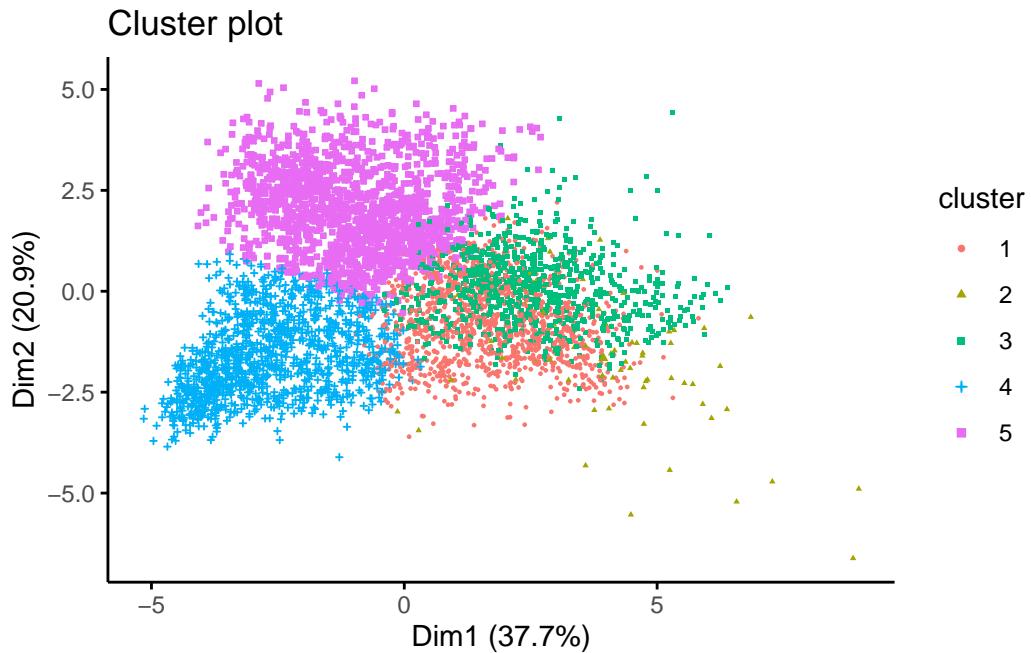
### 3.7.3 Testing

We will evaluate the fit of the k-means model with 5 clusters by creating an  $x$ - $y$  plot 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")
```

```

# Import LSOA boundaries for GLA
st_LSOA <- st_read("./data/geodemographics/LSOA_2011_London_gen_MHW/LSOA_2011_London_gen_MH
W.shp")

Reading layer `LSOA_2011_London_gen_MHW' from data source
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202223/r4ps/data/geodemographi
cshp'
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.

```
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

This chapter illustrates the use of sequence analysis and [WorldPop](#) raster data to identify trajectories of population decline in Ukraine. Developed in Biology for the analysis of DNA sequencing, sequence analysis offers a novel approach to generate a more holistic understanding of population decline trajectories capturing differences in the ordering, frequency, timing and occurrence of population decline. The *longitudinal* and *categorical* nature is a key feature of the data that can be analysed using sequence analysis. In this chapter, We first show how to manipulate gridded, raster population to create a spatial data frame, and explore national and sub-national population trends and spatial structure of population change. We describe the process of implementing sequence analysis to identify trajectories of population decline in a four-stage process. We first define different levels of population change. Second, we apply measure the difference or similarity between individual area-level trajectories. Third, we use an unsupervised machine learning clustering algorithm to identify representative trajectories. Fourth, we use different visualisation tools to understand key features encoded in trajectories.

The chapter is based on the following references:

Gabadinho et al. (2011) describe the functionalities of the `TraMineR` package to visualise and analysis categorical sequence data;

Newsham and Rowe (2022b), González-Leonardo, Newsham, and Rowe (2023) provide empirical applications to identify and study trajectories of population decline across Europe and in Spain;

Tatem (2017) provide an overview of the `WorldPop` data project;

Patias, Rowe, and Arribas-Bel (2021), Patias et al. (2021) provide good examples of the use of sequence analysis to define trajectories of inequality and urban development.

## 4.1 Dependencies

We use the libraries below. Note that to use the `theme_tufte2()` used for `ggplot()` objects in this chapter, you need to call the file `data-viz-themes.R` in the repository.

```

# data manipulation
library(tidyverse)
# spatial data manipulation
library(stars)
library(sf)
# download world pop data
library(wpgpDownloadR) # you may need to install this package running `install.packages("d
# data visualisation
library(viridis)
library(RColorBrewer)
library(patchwork)
library(ggseqplot) # may need to install by running `devtools::install_github("maraab23/gg
# sequence analysis
library(TraMineR)
# cluster analysis
library(cluster)

```

Key packages to this chapter are `TraMineR`, `stars` and `ggseqplot`. `TraMineR` is the go-to package in social sciences for exploring, analysing and rendering sequences based on categorical data. `stars` is designed to handle spatio-temporal data in the form of dense arrays, with space and time as dimensions. `stars` provides classes and methods for reading, manipulating, plotting and writing data cubes. It is a very powerful package. It interacts nicely with `sf` and is suggested to be superior to `raster` and `terra`, which are also known for their capacity to work with multilayer rasters. `stars` is suggested to [deal with more complex data types](#) and [be faster than raster and terra](#). `ggseqplot` provides functionality to visualise categorical sequence data based on `ggplot` capabilities. This differs from `TraMineR` which is based on the base function `plot`. We prefer `ggseqplot` for the wide usage of `ggplot` as a data visualisation tool in R.

## 4.2 Data

The key aim of this chapter is to define representative trajectories of population decline using sequence analysis and WorldPop data. We use WorldPop data for the period extending from 2000 to 2020. WorldPop offers open access gridded population estimates at a high spatial resolution for all countries in the world. WorldPop produces these gridded datasets using a top-down (i.e. disaggregating administrative area counts into smaller grid cells) or bottom-up (i.e. interpolating data from counts from sample locations into grid cells) approach. You can learn about these approaches and the data available from [WorldPop](#).

WorldPop population data are available in various formats:

- Two spatial resolutions: 100m and 1km;

- Constrained and unconstrained counts to built settlement areas;
- Adjusted or unadjusted to United Nations' (UN) national population counts;
- Two formats i.e. `tiff` and `csv` formats.

We use annual 1km gridded, UN adjusted, unconstrained population count data for Ukraine during 2000-2021 in tiff format. We use tiff formats to illustrate the manipulation of raster data. Such skills will come handy if you ever decide to work with satellite imagery or image data in general.

Before calling the data, let's see how we can use `wpgpDownloadR` package. Let's browse the data catalogue.

```
wpgpListCountries() %>%
  head()
```

```
Warning in readLines(con, n = 1): incomplete final line found on
'/var/folders/9z/ql42lpgn22x_c5353k3ycqfr0000gn/T//RtmpHi2m8w/wpgpDatasets.md5'
```

	ISO	ISO3	Country
1	643	RUS	Russia
2	360	IDN	Indonesia
3	840	USA	United States
4	850	VIR	Virgin_Islands_U_S
5	304	GRL	Greenland
6	156	CHN	China

By using the ISO3 country code, let's look for the available datasets for Ukraine.

```
wpgpListCountryDatasets(ISO3 = "UKR") %>%
  head()
```

```
Warning in readLines(con, n = 1): incomplete final line found on
'/var/folders/9z/ql42lpgn22x_c5353k3ycqfr0000gn/T//RtmpHi2m8w/wpgpDatasets.md5'
```

	ISO	ISO3	Country	Covariate
232	804	UKR	Ukraine	ppp_2000
481	804	UKR	Ukraine	ppp_2001
730	804	UKR	Ukraine	ppp_2002
979	804	UKR	Ukraine	ppp_2003
1228	804	UKR	Ukraine	ppp_2004

```
1477 804 UKR Ukraine ppp_2005
```

```
232 Estimated total number of people per grid-cell 2000 The dataset is available to download  
481 Estimated total number of people per grid-cell 2001 The dataset is available to download  
730 Estimated total number of people per grid-cell 2002 The dataset is available to download  
979 Estimated total number of people per grid-cell 2003 The dataset is available to download  
1228 Estimated total number of people per grid-cell 2004 The dataset is available to download  
1477 Estimated total number of people per grid-cell 2005 The dataset is available to download
```

The `wpgpDownloadR` package includes 100m resolution data. To keep things efficient, we use 1km gridded population counts from the [WorldPop data page](#). Obtain population data for Ukraine 2000-2020. We start by reading the set of tiff files using the `read_stars` function from the `star` package.

```
# create a list of file names  
file_list <- fs::dir_ls("./data/sequence-analysis/raster")  
file_list  
  
. /data/sequence-analysis/raster/ukr_ppp_2000_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2001_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2002_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2003_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2004_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2005_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2006_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2007_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2008_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2009_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2010_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2011_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2012_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2013_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2014_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2015_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2016_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2017_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2018_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2019_1km_Aggregated_UNAdj.tif  
. /data/sequence-analysis/raster/ukr_ppp_2020_1km_Aggregated_UNAdj.tif
```

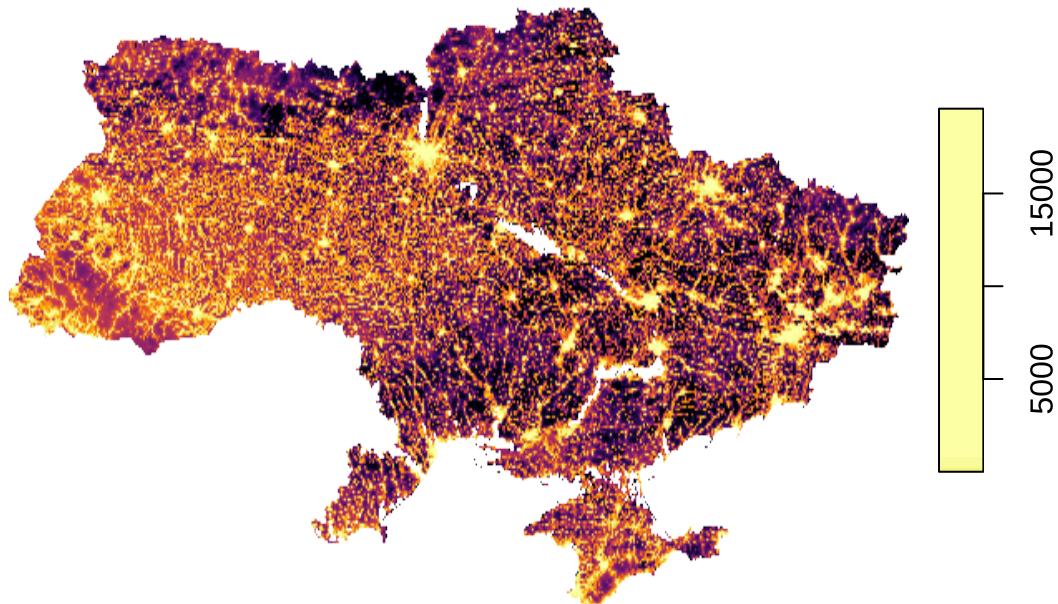
```
# read a list of raster data
pop_raster <- read_stars(file_list, quiet = TRUE)
```

We map the data for 2000 to get a quick understanding of the data.

```
plot(pop_raster[1], col = inferno(100))
```

downsample set to 3

**ukr\_ppp\_2000\_1km\_Aggregated\_UNadj.tif**



Next we read shapefile of administrative boundaries in the form of polygons. We obtain these data from the [GADM website](#). GADM provides maps and spatial data for individuals countries at the national and sub-national administrative divisions. In this chapter, we will work with these data as they come directly from the website which provides a more realistic and similar context to which you will probably come across in the “real-world”.

```
# read spatial data frame
ukr_shp <- st_read("./data/sequence-analysis/ukr_shp/gadm41_UKR_2.shp") %>%
  st_simplify(., # simplify boundaries for efficiency
              preserveTopology = T,
              dTolerance = 1000) %>% # 1km
  sf::st_make_valid(.) %>%
```

```
fortify(.) %>% # turns maps into a data frame so they can more easily be plotted with g  
st_transform(., "EPSG:4326") # set projection system
```

```
Reading layer `gadm41_UKR_2' from data source  
`/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202223/r4ps/data/sequence-anal  
using driver `ESRI Shapefile'  
Simple feature collection with 629 features and 13 fields  
Geometry type: MULTIPOLYGON  
Dimension: XY  
Bounding box: xmin: 22.14045 ymin: 44.38597 xmax: 40.21807 ymax: 52.37503  
Geodetic CRS: WGS 84
```

```
ukr_shp
```

```
Simple feature collection with 629 features and 13 fields (with 1 geometry empty)  
Geometry type: GEOMETRY  
Dimension: XY  
Bounding box: xmin: 22.14519 ymin: 44.38681 xmax: 40.21807 ymax: 52.375  
Geodetic CRS: WGS 84  
First 10 features:  
  GID_2 GID_0 COUNTRY   GID_1   NAME_1 NL_NAME_1           NAME_2  
1      ?    UKR Ukraine     ?      ?      ?          ?  
2  UKR.1.1_1    UKR Ukraine UKR.1_1 Cherkasy            Cherkas'ka  
3  UKR.1.2_1    UKR Ukraine UKR.1_1 Cherkasy            Cherkas'kyi  
4  UKR.1.3_1    UKR Ukraine UKR.1_1 Cherkasy        Chornobaivs'kyi  
5  UKR.1.4_1    UKR Ukraine UKR.1_1 Cherkasy        Chyhyryns'kyi  
6  UKR.1.5_1    UKR Ukraine UKR.1_1 Cherkasy        Drabivs'kyi  
7  UKR.1.6_1    UKR Ukraine UKR.1_1 Cherkasy  Horodyshchens'kyi  
8  UKR.1.7_1    UKR Ukraine UKR.1_1 Cherkasy        Kamians'kyi  
9  UKR.1.8_1    UKR Ukraine UKR.1_1 Cherkasy        Kanivs'ka  
10 UKR.1.9_1    UKR Ukraine UKR.1_1 Cherkasy        Kanivs'kyi  
    VARNAME_2 NL_NAME_2      TYPE_2           ENGTTYPE_2 CC_2  
1          ?       NA      ?          NA      NA  
2          NA       NA Mis'ka Rada City of Regional Significance NA  
3          NA       NA      Raion        District  NA  
4 Chornobayivskyi    NA      Raion        District  NA  
5          NA       NA      Raion        District  NA  
6          NA       NA      Raion        District  NA  
7 Gorodyschenskyi    NA      Raion        District  NA  
8          NA       NA      Raion        District  NA
```

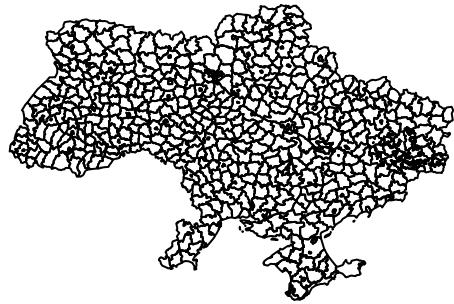
```

9          NA      NA      Misto      City      NA
10         NA      NA      Raion     District    NA
  HASC_2           geometry
1      ? POLYGON ((30.59574 50.40547...
2  UA.CK.CM POLYGON ((32.1715 49.43881, ...
3  UA.CK.CR POLYGON ((32.03393 49.49881...
4  UA.CK.CB POLYGON ((32.17991 49.44486...
5  UA.CK.CY POLYGON ((32.26144 49.20893...
6  UA.CK.DR POLYGON ((32.41852 49.83724...
7  UA.CK.HO POLYGON ((31.56959 49.42509...
8  UA.CK.KN POLYGON ((32.19797 49.20946...
9  UA.CK.KM MULTIPOLYGON (((31.4459 49....
10 UA.CK.KR POLYGON ((31.5851 49.62482, ...

```

Let's have a quick look at the resolution of the administrative areas we will be working. The areas below represent areas at the administrative area level 2 in the spatial data frame `ukr_shp`.

```
plot(ukr_shp$geometry)
```



We ensure that the `pop_raster` object is in the same projection system as `ukr_shp`. So we can make both objects to work together.

```
pop_raster <- st_transform(pop_raster, st_crs(ukr_shp))
```

#### 4.2.1 Data wrangling

For our application, we want to work with administrative areas for three reasons. First, public policy and planning decisions are often made based on administrative areas. These are the areas local governments have jurisdiction, represent and can exert power. Second, migration is a key component of population change and hence directly determines population decline. At a small area, residential mobility may also impact patterns of population potentially adding more complexity and variability to the process. Third, WorldPop data are modelled population estimates with potentially high levels of uncertainty or errors in certain locations. Our aim is to mitigate the potential impacts of these errors.

We therefore recommend working with aggregated data. We aggregate the 1km gridded population data to administrative areas in Ukraine. We use `system.time` to time the duration of the process of aggregation which could take some time depending on your local computational environment.

```
system.time({  
  
  popbyarea_df = aggregate(x = pop_raster,  
                           by = ukr_shp,  
                           FUN = sum,  
                           na.rm = TRUE)  
})  
  
user  system elapsed  
64.352   8.670  73.027
```

#### Sub-national population

The chunk code above returns a list of raster data. We want to create a spatial data frame containing population counts for individual sub-national areas and years. We achieve this by running the following code:

```
# create a function to bind the population data frame to the shapefile  
add_population <- function(x) mutate(ukr_shp,  
                                     population = x)  
  
# obtain sub-national population counts  
ukr_eshp <- lapply(popbyarea_df, add_population)
```

```

# create a dataframe with sub-national populations
select_pop <- function(x) dplyr::select(x, GID_2, NAME_2, population)
population_df <- lapply(ukr_eshp, select_pop) %>%
  do.call(rbind, .)
population_df$year <- rep(seq(2000, 2020), times = 1, each = nrow(ukr_shp))
rownames(population_df) <- rep(seq(1, nrow(population_df)), by=1), times = 1

# sub-national spatial data frame
population_df

```

```

Simple feature collection with 13209 features and 4 fields (with 21 geometries empty)
Geometry type: GEOMETRY
Dimension:      XY
Bounding box:   xmin: 22.14519 ymin: 44.38681 xmax: 40.21807 ymax: 52.375
Geodetic CRS:   WGS 84
First 10 features:
  GID_2          NAME_2 population           geometry year
1     ?      301849.00 POLYGON ((30.59574 50.40547...
2  UKR.1.1_1    Cherkas'ka  280917.39 POLYGON ((32.1715 49.43881,...
3  UKR.1.2_1    Cherkas'kyi  89116.78 POLYGON ((32.03393 49.49881...
4  UKR.1.3_1  Chornobaivs'kyi  50096.24 POLYGON ((32.17991 49.44486...
5  UKR.1.4_1    Chyhyryns'kyi  36646.73 POLYGON ((32.26144 49.20893...
6  UKR.1.5_1    Drabivs'kyi  42467.86 POLYGON ((32.41852 49.83724...
7  UKR.1.6_1 Horodyshchens'kyi  49886.59 POLYGON ((31.56959 49.42509...
8  UKR.1.7_1    Kamians'kyi  35587.28 POLYGON ((32.19797 49.20946...
9  UKR.1.8_1    Kanivs'ka   14406.93 MULTIPOLYGON (((31.4459 49....
10 UKR.1.9_1   Kanivs'kyi  37495.04 POLYGON ((31.5851 49.62482,... 2000

```

## National population

We also create a data frame providing population counts at the national level.

```

# obtain national population counts
population_count <- map_dbl(ukr_eshp, ~.x %>%
  pull(population) %>%
  sum(na.rm = TRUE)
) %>%
  as.data.frame()

# change labels
colnames(population_count) <- c("population")

```

```

rownames(population_count) <- rep(seq(2000, 2020, by=1), times = 1)
population_count$year <- rep(seq(2000, 2020, by=1), times = 1)

# national annual population counts
population_count

```

	population	year
2000	47955683	2000
2001	47520197	2001
2002	47094225	2002
2003	46700872	2003
2004	46330322	2004
2005	46011048	2005
2006	45734099	2006
2007	45502336	2007
2008	45286748	2008
2009	45090608	2009
2010	44923112	2010
2011	44744969	2011
2012	44593427	2012
2013	44424702	2013
2014	44250993	2014
2015	44068072	2015
2016	43856852	2016
2017	43622605	2017
2018	43391259	2018
2019	43140679	2019
2020	42880388	2020

### 4.2.2 Exploratory data analysis

Now we are ready to start analysing the data. Before building complexity on our analysis, conducting some exploratory data analysis to understand the data is generally a good starting point, particularly given the multi-layer nature of the data at hand - capturing space, time and population levels.

#### National patterns

We first analyse national population trends. We want to know to what extent the population of Ukraine has declined over time over the last 20 years. An effective way to do this is to compute summary statistics and visualise the data. Below we look at year-to-year changes in

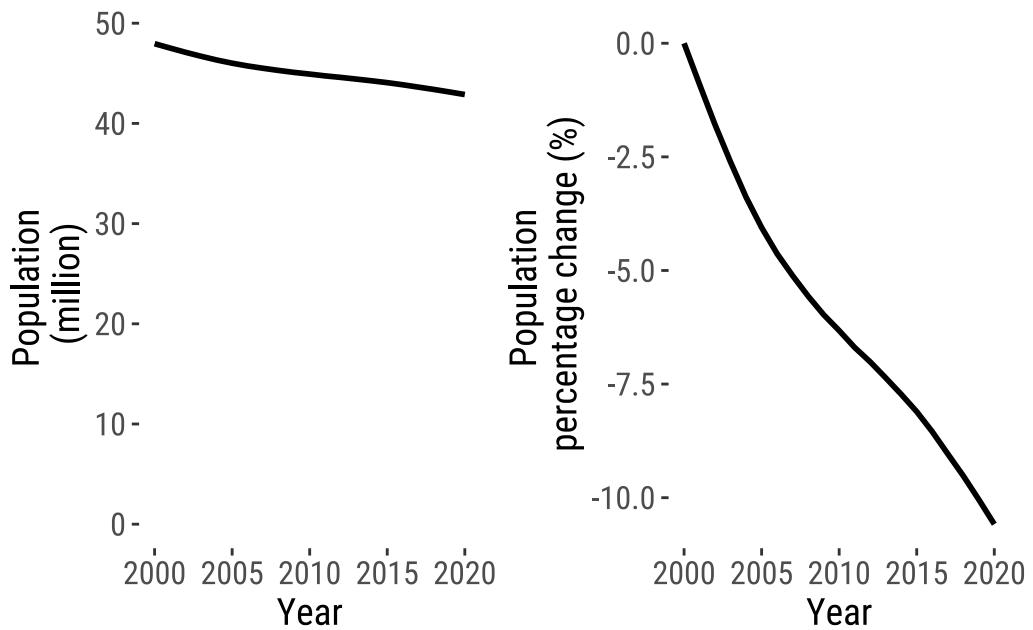
population levels and as a percentage change. By using `patchwork`, we combine two plots into a single figure.

```
# visualise national population trends
pop_level_p <- ggplot(population_count,
  aes(x = year, y = population/1000000 )) +
  geom_line(size = 1) +
  theme_tufte2() +
  ylim(0, 48) +
  labs(y = "Population \n(million)",
       x = "Year")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
  i Please use `linewidth` instead.
```

```
# visualise percentage change in population
pop_percent_p <- population_count %>%
  mutate(
    pct_change = ( ( population - 47955683) / 47955683) * 100
  ) %>%
  ggplot(aes(x = year, y = pct_change)) +
  geom_line(size = 1) +
  theme_tufte2() +
  labs(y = "Population \npercentage change (%)",
       x = "Year")

pop_level_p | pop_percent_p
```



### Sub-national

Population losses are likely vary across the country. From previous research we know that rural and less well connected areas tend to lose population through the internal migration of young individuals as they move for work and job opportunities (Rowe, Corcoran, and Bell 2016). We also know that they tend to move to large, densely populated cities where these opportunities are concentrated and because they also offer a wide variety of amenities and activities. Cities tend to work as accelerators enabling fast career development and occupational progression (Fielding 1992). Though, we have also seen the shrinkage of populations in cities, particularly in eastern European countries (Turok and Mykhnenko 2007).

To examine the patterns of sun-national population losses, we compute two summary measures: (1) annual percentage change in population; and, (2) overall percentage change in population between 2000 and 2021. We start by looking the overall percentage change as it is easier to visualise. To this end, we categorise our measure of overall percentage change into seven different classes. Based on previous work by González-Leonardo, Newsham, and Rowe (2023), we classify changes into high decline ( $\leq -3$ ), decline ( $> -3$  and  $\leq -1.5$ ), moderate decline ( $> -1.5$  and  $\leq -0.3$ ), stable ( $< -0.3$  and  $< 0.3$ ), moderate growth ( $\geq 0.3$  and  $< 1.5$ ), growth ( $\geq 1.5$  and  $< 3$ ) and high growth ( $\geq 3$ ). Let's first create the measures of population change.

```
# compute population change metrics
population_df <- population_df %>%
  dplyr::group_by(GID_2) %>%
```

```

arrange(-year, .by_group = TRUE ) %>%
mutate(
  pct_change = ( population / lead(population) - 1) * 100, # rate of population change
  pct_change_2000_21 = ( population[year == "2020"] / population[year == "2000"] - 1) * 100,
  ave_pct_change_2000_21 = mean(pct_change, na.rm = TRUE)
) %>%
ungroup()

```

Let's map the overall percentage change in population between 2000 and 2020. We see a wide spread pattern of population decline across Ukraine. We observe a large spatial cluster of high population decline across the country with moderate population decline in some areas. Administrative areas containing large cities seem to record overall population growth between 2000 and 2020, potentially absorbing population movements from the rest of the country. What else do you think may be driving population growth in cities? And in contrast, what do you think is contributing to population decline in most of Ukraine?

```

# set colours
cols <- c("#7f3b08", "#b35806", "#e08214", "#faf0e6", "#8073ac", "#542788", "#2d004b")
# reverse order
cols <- rev(cols)

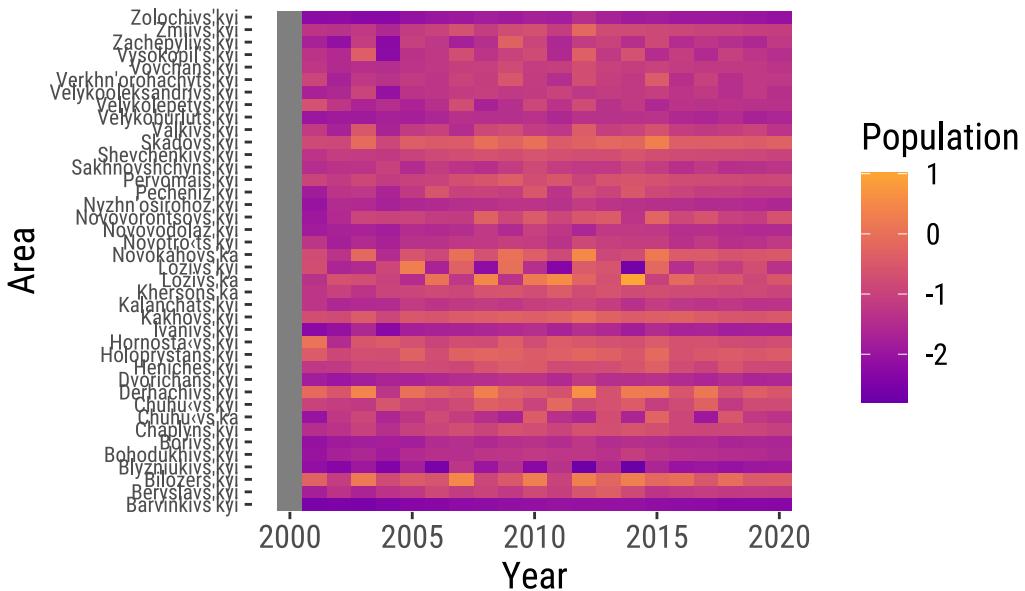
population_df %>% dplyr::filter( year == 2020) %>%
  drop_na(pct_change_2000_21) %>%
  mutate(
    ove_pop_class = case_when( pct_change_2000_21 <= -3 ~ 'high_decline',
                                pct_change_2000_21 <= -1.5 & pct_change_2000_21 > -3 ~ 'decline',
                                pct_change_2000_21 <= -.3 & pct_change_2000_21 > -1.5 ~ 'moderate',
                                pct_change_2000_21 > -0.3 & pct_change_2000_21 < 0.3 ~ 'stable',
                                pct_change_2000_21 >= 0.3 & pct_change_2000_21 < 1.5 ~ 'moderate',
                                pct_change_2000_21 >= 1.5 & pct_change_2000_21 < 3 ~ 'growth',
                                pct_change_2000_21 >= 3 ~ 'high_growth'),
    ove_pop_class = factor(ove_pop_class,
                           levels = c("high_decline", "decline", "moderate_decline", "stable", "moderate_gro
    ) %>%
  ggplot(aes(fill = ove_pop_class)) +
  geom_sf(col = "white", size = .1) +
  scale_fill_manual(values = cols,
                    name = "Population change") +
  theme_map_tufte()

```



Now that we have understanding of population changes over the whole 2000-2020 period. Let's try to understand how different places arrive to different outcomes. A way to do this is to look at the evolution of population changes. Different trajectories of population change could underpin the outcomes of population change that we observe today. Current outcomes could be the result of a consistent pattern of population decline over the last 20 years. They could be the result of acceleration in population loss after a major natural or war event, or they could reflect a gradual process of erosion. We visualise way to get an understanding of this is to analyse annual percentage population changes across individual areas. We use a Hovmöller Plot as illustrated by Rowe and Arribas-Bel (2022) for the analysis of spatio-temporal data.

```
population_df %>% dplyr::filter( ave_pct_change_2000_21 < 0) %>%
  tail(., 40*21) %>%
  ggplot(data = .,
         mapping = aes(x= year, y= reorder(NAME_2, pct_change), fill= pct_change)) +
  geom_tile() +
  scale_fill_viridis(name="Population", option ="plasma", begin = .2, end = .8, direction
  theme_tufte2() +
  labs(title= paste(" "), x="Year", y="Area") +
  theme(text = element_text(size=14)) +
  theme(axis.text.y = element_text(size=8))
```



The Hovmöller Plot shows that most of the selected areas tend to experience annual population decline, with varying spells of population growth. Percentage population changes range between 1 and -2.5. We also observe areas with consistent trajectories of annual population decline, like Zolochivs'kyi and Barvinkivs'kyi, and areas with strong decline in the first few years between 2000 and 2005 but moderate decline later on, such as Novovorontsovsk'kyi. Yet, Hovmöller Plots provide a limited understanding of the annual population changes for a handful of areas at the time and it is therefore difficult to identify systematic representative patterns. Here we have selected 40 areas of a total of 629. Displaying the total number of areas in a Hovmöller Plot will not produce readable results. Even if that was the case, it would be difficult to identify systematic patterns. As we will seek to persuade you below, sequence analysis provides a very novel way to define representative trajectories in the data, identify systematic patterns and extract distinctive features characterising those trajectories.

### 4.3 Application

Next, we focus on the application of sequence analysis to identify representative trajectories of population decline at the sub-national level between 2000 and 2020 in Ukraine. Intuitively, sequence analysis can be seen as a four-stage process. First, it requires the definition of longitudinal categorical outcome. Second, it measures the dissimilarity of individual sequences via a process known as optimal matching (OM). Third, it uses these dissimilarity measures to define

a typology of representative trajectories using unsupervised machine learning clustering techniques. Fourth, trajectories can be visualised and their distinctive features can be measured. Below we describe the implementation of each stage to identify representative trajectories of population decline.

#### 4.3.1 Defining outcome process

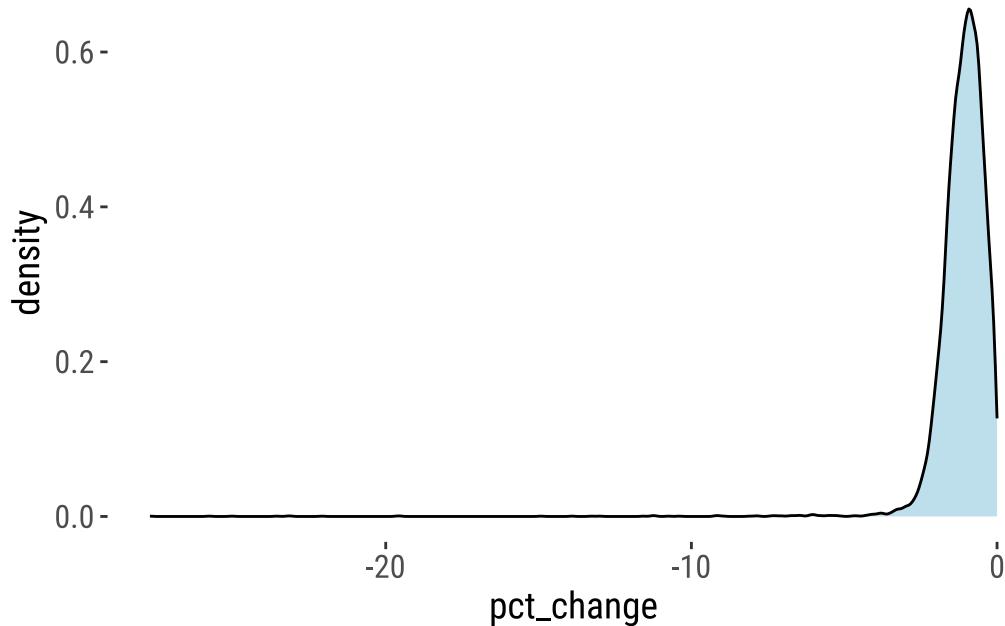
Sequence analysis requires longitudinal categorical data as an input. We therefore classify our population count data into distinct categorical categories, henceforth referred to as *states* of population change. We compute the annual percentage rate of population change for individual areas and use these rates to measure the extent and pace of population change. The annual rate of population change is computed as follows:

$$\frac{p(t1) - p(t0)}{p(t0)} * 100$$

where:  $p(t0)$  is the population at year  $t0$  and  $p(t1)$  is the population at  $t + 1$ .

As previously, we differentiate areas of high decline, decline, moderate decline, stable, moderate growth, growth and high growth. For the analysis, we focus on areas recording population losses between 2000 and 2020. The histogram shows the magnitude and distribution of population decline over this period. We observe that most occurrences of decline are moderate around zero, while very few exceed 5%.

```
# select areas reporting losses between 2000 and 2020
population_loss_df <- population_df %>%
  dplyr::filter( pct_change_2000_21 < 0)
# plot distribution of percentage change
population_loss_df %>%
  dplyr::filter(pct_change < 0) %>%
  ggplot(data = ) +
  geom_density(alpha=0.8, colour="black", fill="lightblue", aes(x = pct_change)) +
  theme_tufte()
```



Next we classify the annual percentage of population change into our seven states.

```
# remove 2000 as it has no observations of population change
population_loss_df <- population_loss_df %>%
  dplyr::filter( year != 2000)
# clasify data
population_loss_df <- population_loss_df %>%
  mutate(
    pop_class = case_when( pct_change <= -3 ~ 'high_decline',
                           pct_change <= -1.5 & pct_change > -3 ~ 'decline',
                           pct_change <= -.3 & pct_change > -1.5 ~ 'moderate_decline',
                           pct_change > -0.3 & pct_change < 0.3 ~ 'stable',
                           pct_change >= 0.3 & pct_change < 1.5 ~ 'moderate_growth',
                           pct_change >= 1.5 & pct_change < 3 ~ 'growth',
                           pct_change >= 3 ~ 'high_growth')
  )
```

### 4.3.2 Optimal matching

We measure the extent of dissimilarity between individual sequence of population decline. To this end, we used a sequence analysis technique, OM, which computes distances between sequences as a function of the number of transformations required to make sequences identical.

Two sets of operations are generally used: (1) insertion/deletion (known as indel) and (2) substitution operations. Both of these operations represent the cost of transforming one sequence into another. These costs are challenging to define and below we discuss what is generally used in empirical work. Intuitively, the idea of OM is to estimate the cost of transforming one sequence into another so that the greater the cost to make two sequences identical, the greater the dissimilarity and *vice versa*.

*Indel operations* involve the addition or removal of an element within the sequence and substitution operations are the replacement of one element for another. Each of these operations is assigned a cost, and the distance between two sequences is defined as the minimum cost to transform one sequence to another (Abbott and Tsay 2000). By default, indel costs are set to 1. To illustrate indel operations, let's consider an example of sequences of annual population change for three areas during 2000 and 2003. The sequences are identical, except for 2003. In this case, indel operations involve the cost of transforming the status *stable* in the sequence for area 1 to *high decline* in the sequence for area 2, and thus this operation would return a cost of 2. Why 2? It is 2 because you would need to delete *stable* and add *high decline*. Now, let's try the cost of transforming the status *stable* in the sequence for area 1 to the status in the sequence for area 3 using indel operations. What is the cost? The answer is 1 because we only need to delete *stable* to make it identical.

Area	2000	2001	2002	2003
1	decline	decline	decline	stable
2	decline	decline	decline	high decline
3	decline	decline	decline	-

*Substitution operations* or costs represent transition costs; that is, the cost for substituting each state with another. Substitution costs are defined in one of two ways (Salmela-Aro et al. 2011). One approach is the theory-driven approach. In such approach, substitution costs are grounded in theory suggesting that, for example, transforming state 1 to state 2 should have a greater cost than transforming state 1 to state 3, or performing the opposite operation i.e. transforming state 2 to state 1. An example could be that it is more financially costly to transition from *full-time employment* to *full-time education* than transition from *full-time education* to *full-time employment*.

A second approach and most commonly used in empirical work is a data-driven approach. In this approach, substitution costs are empirically derived from transition rates between states. The cost of substitution is inversely related to the frequency of observed transitions within the data. This means that infrequent transitions between states have a higher substitution cost. For example, as we will see, transitions from the state of high decline to high growth are rarer than from high growth to high decline in Ukraine. The transition rate between state  $i$  and state  $j$  is the probability of observing state  $j$  at time  $t_1$  given that the state  $i$  is observed at time  $t$  for  $i \neq j$ . The substitution cost between states  $i$  and  $j$  is computed as:

$$2 - p(i|j) - p(j|i)$$

where  $p(i|j)$  is the transition rate between state  $i$  and  $j$ .

To implement OM, we first need to rearrange the structure of our data from long to wide format. You can now see now that individual rows represent areas (column 1) and columns from 2 to 21 represent years.

- see Rowe and Arribas-Bel (2022) for a description on different spatio-temporal data structures and their manipulation using tidyverse principles.

```
# transform from long to wide format
wide_population_loss_df <- population_loss_df %>%
  as_tibble() %>%
  group_by(GID_2) %>%
  arrange(year, .by_group = TRUE ) %>%
  ungroup() %>%
  tidyr::pivot_wider(
    id_cols = GID_2,
    names_from = "year",
    values_from = "pop_class"
  )

wide_population_loss_df

# A tibble: 571 x 21
# ... with 561 more rows, and 10 more variables: `2011` <chr>, `2012` <chr>,
#   `2013` <chr>, `2014` <chr>, `2015` <chr>, `2016` <chr>, `2017` <chr>,
#   `2018` <chr>, `2019` <chr>, `2020` <chr>
```

Once the data frame has been reshaped into a wide format, we define the data as a state sequence object using the R package **TraMineR**. Key here is to appropriately define the labels and an appropriate palette of colours. Depending on the patterns you are seeking to capture a diverging, sequential or qualitative colour palette may be more appropriate. For this chapter, we use a diverging colour palette as we want to effectively represent areas experiencing diverging patterns of population decline or growth.

Note: various types of sequence data representation exist in **TraMineR**. These representations vary in the way they capture states or events. Chapter 4 in Gabadinho et al. (2009) describes the various representations that **TraMineR** can handle. In any case, the state sequence representation used in this chapter is the most commonly used and internal format used by **TraMineR**. Hence we focus on it.

```
# alphabet
seq.alphab <- c("high_growth", "growth", "moderate_growth", "stable", "moderate_decline",
# labels
seq.lab <- c("High growth", "Growth", "Moderate growth", "Stable", "Moderate decline", "De
# define state sequence object
seq.cl <- seqdef(wide_population_loss_df,
                  2:21,
                  alphabet = seq.alphab,
                  labels = seq.lab,
                  cnames = c("2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008")
                  cpal =c("1" = "#7f3b08",
                         "2" = "#b35806",
                         "3" = "#e08214",
                         "4" = "#faf0e6",
                         "5" = "#8073ac",
                         "6" = "#542788",
                         "7" = "#2d004b"))
```

[>] 7 distinct states appear in the data:

```
1 = decline

2 = growth

3 = high_decline

4 = high_growth
```

```

5 = moderate_decline

6 = moderate_growth

7 = stable

[>] state coding:

[alphabet]          [label]          [long label]

1 high_growth       high_growth     High growth
2 growth            growth          Growth
3 moderate_growth   moderate_growth Moderate growth
4 stable             stable          Stable
5 moderate_decline moderate_decline Moderate decline
6 decline            decline          Decline
7 high_decline      high_decline    High decline

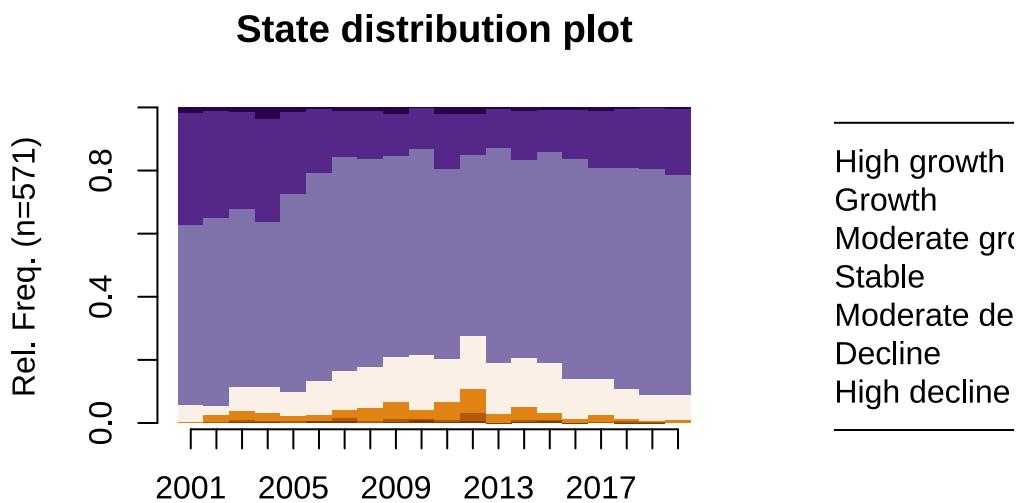
[>] 571 sequences in the data set

[>] min/max sequence length: 20/20

```

Using the sequence data object, we create a state distribution plot to get an understanding of the data. The plot shows the distribution of areas across status of population change in individual years. The overall picture emerging from the plot is an overall pattern of population decline between 2000 and 2020, predominantly moderate decline and limited spells of high population decline or growth. This aligns with the predominant trajectory of population decline observed in Ukraine based on more aggregate data at the regional level (Newsham and Rowe 2022a).

```
seqplot(seq.cl,
        title="State distribution plot",
        type = "d",
        with.legend = "right",
        border = NA)
```



We now move on to compute the substitution costs for our population states. From the equation above, you may have realised that transition costs vary between 0 and 2, with the former indicating zero cost. The latter indicates the maximum cost of converting one state into another. The option `TRATE` in method states to derive costs from observed transition rates. That is the data-driven approach discussed above. From the matrix below, you can see that it is more costly to convert a status “high growth” to “moderate decline” than from “growth” to “moderate”. This makes sense. We expect gradual changes in population along a given trajectory if they were to occur due to natural causes.

Note we are considering a fixed measure of transition rates. That means that we are using the whole dataset to compute an average transition rate between states. That assumes that the rate of change between states does not change over time. Yet, there may be good reasons to believe they do as areas move across different states. In empirical work, time varying transition rates are more often considered. That means we use temporal slices of the data to compute transition rates; for example, using data from 2001, 2002 and so on. In this way, we end up with

potentially different transition rates for every year.

```
# Calculate transition rates
subs_costs <- seqsubm(seq.cl,
                      method = "TRATE",
                      #time.varying = TRUE
                      )
```

```
subs_costs
```

	high_growth	growth	moderate_growth	stable	moderate_decline
high_growth	0.000000	1.975000	1.940066	1.949203	1.949277
growth	1.975000	0.000000	1.867730	1.867946	1.634772
moderate_growth	1.940066	1.867730	0.000000	1.709425	1.547832
stable	1.949203	1.867946	1.709425	0.000000	1.501723
moderate_decline	1.949277	1.634772	1.547832	1.501723	0.000000
decline	1.847681	1.848052	1.826810	1.915740	1.643018
high_decline	1.288462	1.700000	1.773408	1.917223	1.805088
	decline	high_decline			
high_growth	1.847681	1.288462			
growth	1.848052	1.700000			
moderate_growth	1.826810	1.773408			
stable	1.915740	1.917223			
moderate_decline	1.643018	1.805088			
decline	0.000000	1.844570			
high_decline	1.844570	0.000000			

To understand better the idea of substitution costs, we can have direct look at transition rates underpinning these costs. Transition rates can be computed via `seqrate`. By definition, transition rates vary between 0 and 1, with zero indicating no probability of a transition occurring. One indicates a 100% probability of a transition taking place. Thus, for example, the matrix below tell us that there is a 5% probability of observing a transition from “high growth” to “moderate decline” in our sample. Examining transition rates could provide very valuable information about the process in analysis.

```
seq.rate <- seqrate(seq.cl)
```

```
[>] computing transition probabilities for states high_growth/growth/moderate_growth/stable,
```

```
round(seq.rate, 2)
```

	[→ high_growth]	[→ growth]	[→ moderate_growth]
[high_growth ->]	0.12	0.03	0.05
[growth ->]	0.00	0.05	0.11
[moderate_growth ->]	0.01	0.02	0.09
[stable ->]	0.00	0.00	0.04
[moderate_decline ->]	0.00	0.00	0.02
[decline ->]	0.00	0.01	0.02
[high_decline ->]	0.16	0.10	0.18
	[→ stable]	[→ moderate_decline]	[→ decline]
[high_growth ->]	0.05	0.05	0.15
[growth ->]	0.13	0.36	0.15
[moderate_growth ->]	0.25	0.43	0.15
[stable ->]	0.46	0.42	0.05
[moderate_decline ->]	0.08	0.82	0.08
[decline ->]	0.03	0.28	0.65
[high_decline ->]	0.07	0.19	0.15
	[→ high_decline]		
[high_growth ->]	0.55		
[growth ->]	0.20		
[moderate_growth ->]	0.05		
[stable ->]	0.01		
[moderate_decline ->]	0.00		
[decline ->]	0.01		
[high_decline ->]	0.15		

Now we focus on the probably most important component of sequence analysis; that is, the calculation of dissimilarity. Recall our aim is to identify representative trajectories. To this end, we need a way to measure how similar or different sequences are - which is known as OM. Above, we described that we can use indel and substitution operations to measure the dissimilarity or costs between individual sequences. The code chunk implements OM based on indel and substitution operations. The algorithm takes an individual sequence and compares it with all of the sequences in the dataset, and identifies the sequence with the minimum cost i.e. the most similar sequence. The result of this computing intensive process is a distance matrix encoding the similarity or dissimilarity between individual sequences.

For indel, auto sets the indel as `max(sm)/2` when sm is a matrix. For more details, run `?seqdist` on your console

```
# Calculate a distance matrix
seq.om <- seqdist(seq.cl,
                  method = "OM", # specify the method
                  indel = "auto", # specify indel costs
```

```
sm = subs_costs) # specify substitution costs
```

As highlighted above, if you would like to apply varying substitution costs, you can do this directly here by using the option `method = DHD`.

### 4.3.3 Clustering

The resulting distance matrix from OM `seq.om` indicates the degree of similarity between individual sequences. To identify representative trajectories, we then need to a way to group together similar sequences to produce a typology, in this case of population decline trajectories. Unsupervised cluster analysis is generally used for this task. Trusting you have built an understanding of cluster analysis from the previous chapter, we will not provide an elaborate description here. If you would like to know more about cluster analysis, we recommend the introductory book by Kaufman and Rousseeuw (2009). We use a clustering method called `k-medoids`. This methods is known to be more robust to noise and outliers than the conventional k-means procedure (Backman, Lopez, and Rowe 2020). This is because the medoid algorithm clusters the data by minimising a sum of pair-wise dissimilarities (Kaufman and Rousseeuw 2009), rather than a sum of squared Euclidean distances. We run cluster analyses at different numbers of `k` starting from 2 to 20.

```
# run PAMs
for (k in 2:20)
  pam_sol <- pam(seq.om, k)
```

We then seek to determine the optimal number of clusters `k`. We use silhouette scores, but as we noted Chapter 3, the optimal number of clusters is better determined by the user given the context and use case. It is an art. There is no wrong or right answer. As can be seen from the results below from the average silhouette score, two clusters is suggested as the optimal solution. However, we could argue that we gain very little from such coarse partition of the data. We suggest to take this as guidance and a starting point to look to identify an appropriate data partition. We suggest to visualise different solution and gain an understanding of what data get split and decide on whether the resulting patterns contribute to the understanding of the process at hand.

```
# compute average silhouette scores for all 20 cluster solutions
asw <- numeric(20)
for (k in 2:20)
  asw[k] <- pam(seq.om, k) $ silinfo $ avg.width
k.best <- which.max(asw)
cat("silhouette-optimal number of clusters:", k.best, "\n")
```

```
silhouette-optimal number of clusters: 2
```

```
asw
```

```
[1] 0.0000000 0.5729062 0.5363097 0.4923203 0.4629785 0.4425575 0.4417958  
[8] 0.4559577 0.4597152 0.4569222 0.4682526 0.4832332 0.4359832 0.4365490  
[15] 0.4249453 0.4193369 0.4317815 0.4323037 0.4333972 0.4463586
```

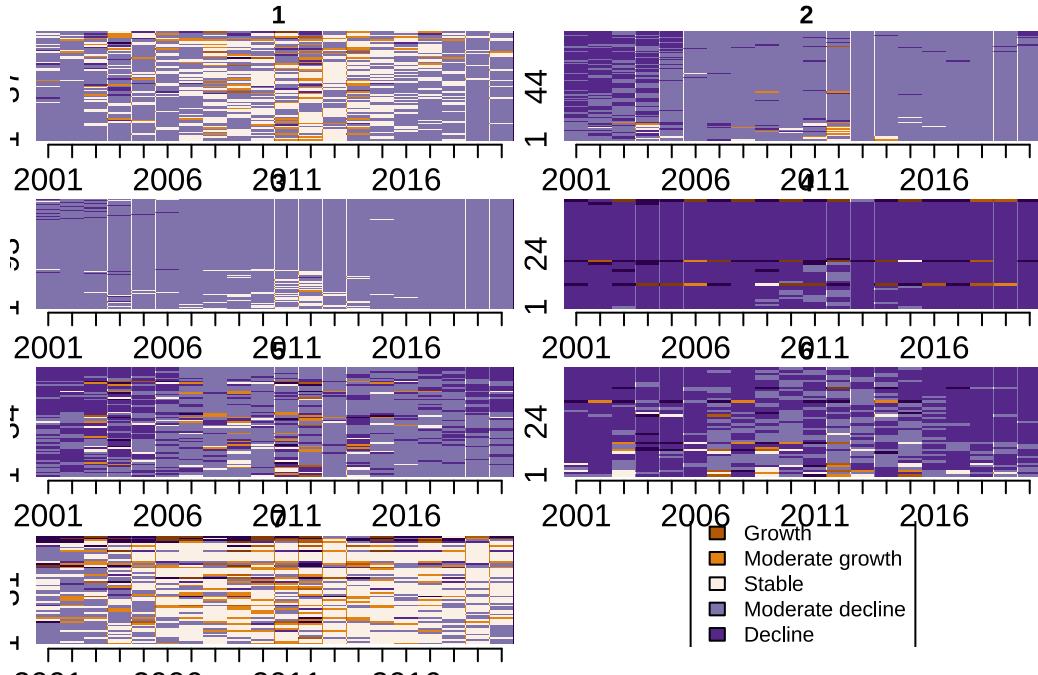
We rerun and save the results for a 7k cluster partition. If you inspect the resulting data frame, it provides an identifier for each cluster. Each individual area is attributed to a cluster. Next the question that we seek to answer is what sort of pattern do these clusters capture?

```
# rerun pam for k=7  
pam_optimal <- pam(seq.om, 7)
```

#### 4.3.4 Visualising

To understand the representative patterns captured in our data partition, we use visualisation. There is a battery of different visualisation tools to extract information and identify distinctive features of the identified trajectories. We start by using *individual sequence plots* by trajectory type. They provide a visual representation of how individual areas in each trajectory type moves between states. Recall that we are capturing representative trajectories; hence, there is still quite a bit of variability in terms of the patterns encapsulated in each representative trajectory. Back to the individual sequence plots, each line in these plots represents an area. Time is displayed horizontally and colours encode different states - in our case of population change. Numbers on the y-axis display the number of areas in each cluster. The figure immediate below relies on the base `plot` library, and by default, it is not very visually appealing.

```
# create individual sequence plots  
par(mar=c(1,1,1,1))  
seqplot(seq.cl,  
        group = pam_optimal$clustering,  
        type = "I",  
        border = NA,  
        cex.axis = 1.5,  
        cex.lab = 1.5,  
        sortv = seq.om)
```



We therefore switch to the R library `ggseqplot` which enables visualisation of sequence data based on `ggplot` functionalities. This package may provide more flexibility if we are more familiar with `ggplot`.

The figure below offers a clear representation of the systematic sequencing of states that each trajectory captures. It provides information on two key features of trajectories: sequencing and size. For example, trajectory 1 seems to capture a sequencing pattern of transitions from moderate population decline to stability and back to moderate decline. Trajectory 2 shows a pattern high population decline during the first few years and then consistent moderate decline. Trajectory 3 displays a predominant pattern of moderate population decline. Trajectory 4 represents patterns of areas experiencing decline with spells of high population decline. Trajectory 5 shows a pattern of decline in the first few years followed by moderate decline and decline again. Trajectory 6 shows a similar pattern with more prevalent spells of population decline across the entire period. Trajectory 7 displays a trend of temporary decline, with spells of population growth and stability. From these plots, you can also identify which trajectories tend to be more common. In our example, trajectory 3 accounts the largest number of areas: 189.

```
# create individual sequence plots based on ggplot
ggseqiplot(seq.cl,
  group = pam_optimal$clustering,
  sortv = seq.om,
```

```

    facet_ncol = 4) +
scale_fill_manual(values = rev(cols)) +
scale_color_manual(values = rev(cols))

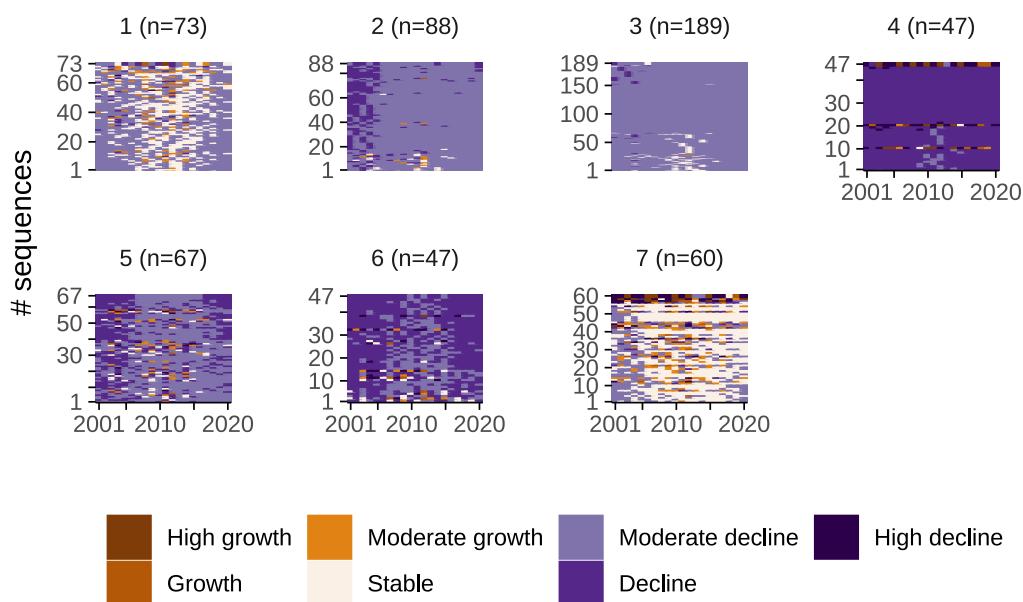
```

Scale for fill is already present.

Adding another scale for fill, which will replace the existing scale.

Scale for colour is already present.

Adding another scale for colour, which will replace the existing scale.



We can also get a better understanding of the resulting trajectories by analysing *state frequency plots*. They show the number of occurrences of a given state in individual years. These plots examine the data from a vertical perspective i.e. looking at individual years across areas, rather than at individual areas over time. State frequency plots reveal that predominant states in each year and changes in their prevalence. Focusing on trajectory 1, for example, we observe that moderate decline was the predominant state between 2000 and 2007 and stability became equally prevalent during 2008 and 2015.

```

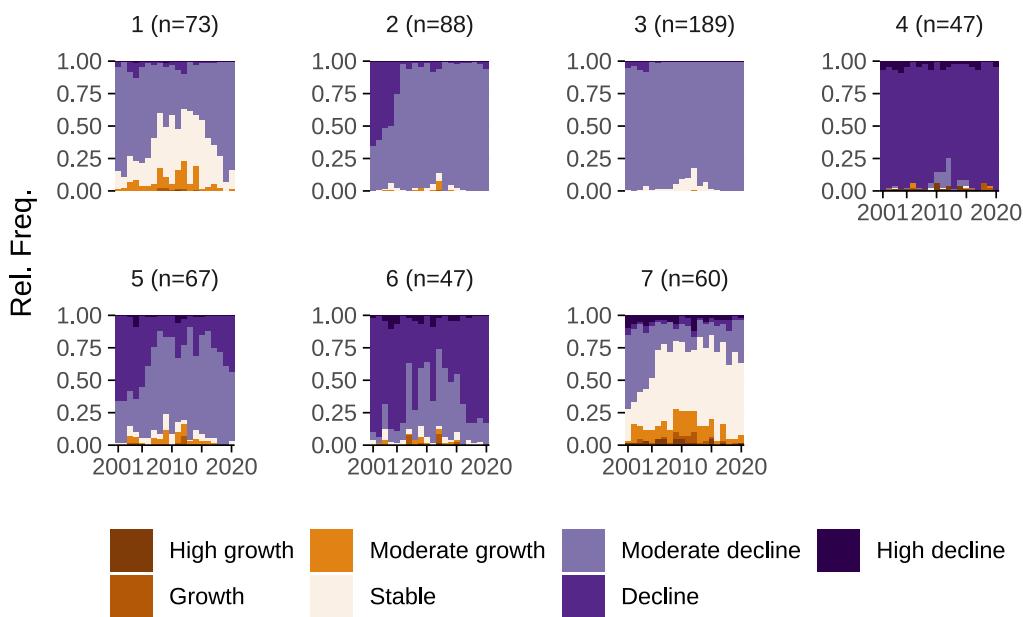
# create state frequency plots based on ggplot
ggseqdplot(seq.cl,
  group = pam_optimal$clustering,
  facet_ncol = 4) +

```

```
scale_fill_manual(values = cols) +
scale_color_manual(values = cols)
```

Scale for fill is already present.

Adding another scale for fill, which will replace the existing scale.

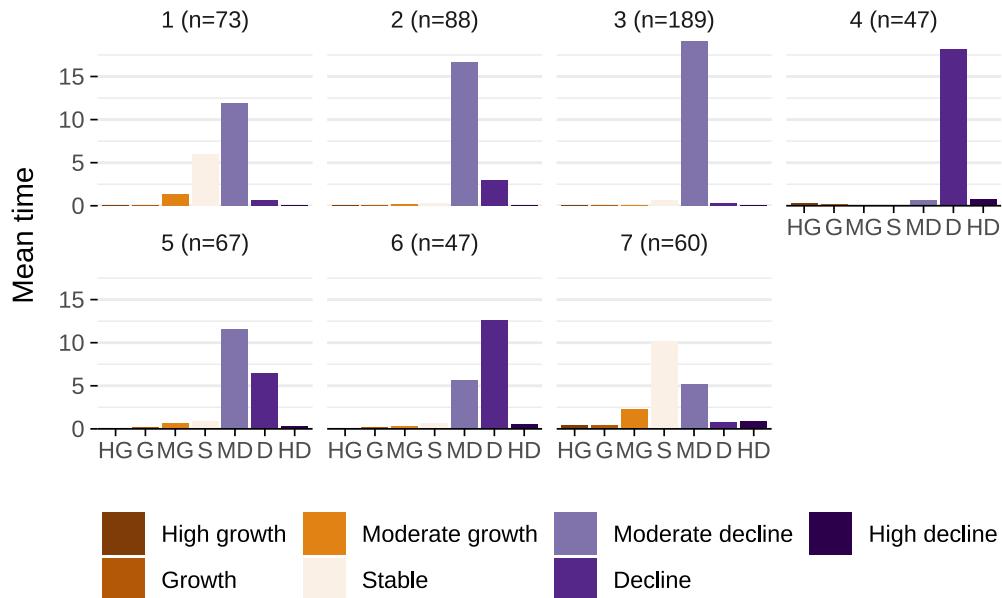


We can also examine time spent in individual states in each trajectory. *Time spent plots* report the average time spent in each state. The measure of time depends on the original data used in the analysis. We use years so the y-axis refers to the average number of years that a given status appears in a representative trajectory type. For example, a score over 5 for stable in trajectory 1 indicates that the average number of years that areas in that typology are classified in that category is over 5.

```
# create time spent plots based on ggplot
ggseqmtpplot(seq.cl,
  group = pam_optimal$clustering,
  facet_ncol = 4) +
  scale_fill_manual(values = rev(cols)) +
  scale_color_manual(values = rev(cols)) +
  scale_x_discrete(labels=c("high_growth" = "HG", "growth" = "G",
  "moderate_growth" = "MG", "stable" = "S", "moderate_decline"
```

Scale for fill is already present.

Adding another scale for fill, which will replace the existing scale.

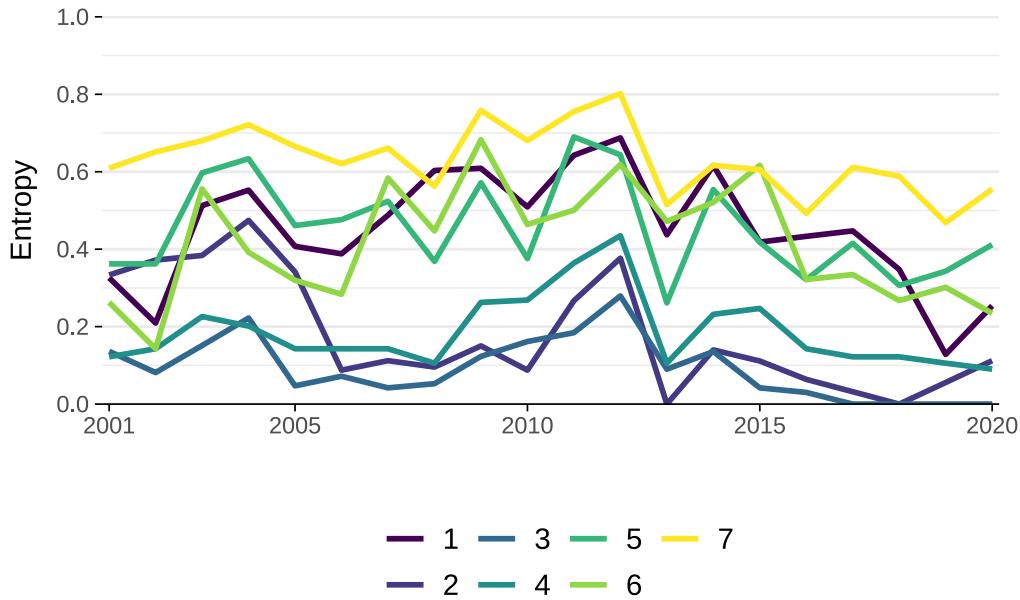


Finally we analyse *entropy index plots*. Entropy is a measure of diversity. The greater the score, the greater the entropy or diversity of states. The plot below displays the entropy index computed for individual trajectories each year. Each line represents the entropy index for a trajectory in each year. The top yellow line in 2001 indicates that in 2001 areas following a trajectory 7 type were distributed across a larger number of states than any other trajectory, reflecting the fact that areas experiencing this trajectory moves between states of decline, stability and growth. In other word, it indicates that there was more diversity of states.

```
# create entropy index plots based on ggplot
ggseqeplot(seq.cl,
  group = pam_optimal$clustering) +
  scale_colour_viridis_d()
```

Scale for colour is already present.

Adding another scale for colour, which will replace the existing scale.



## 4.4 Questions

For the first assignment, we will continue to focus on London as our area of analysis. We will use population count estimates from the Office of National Statistics (ONS). The dataset provides information on area, population numbers and population density at national, regional and smaller sub-national area level, including Unitary Authority, Metropolitan County, Metropolitan District, County, Non-metropolitan District, London Borough, Council Area and Local Government District for the period from 2001 to 2020.

```
pop_df <- read_csv("./data/sequence-analysis/population_uk/population-uk-2011_20.csv")
pop_df
```

```
# A tibble: 420 x 44
  Code     Name Geogr~1 Area ~2 Estim~3 2020 ~4 Estim~5 2019 ~6 Estim~7 2018 ~8
  <chr>   <chr> <chr>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1 K02000~ UNIT~ Country  2.43e5  6.71e7  276.   6.68e7  275.   6.64e7
2 K03000~ GREA~ Country  2.29e5  6.52e7  285.   6.49e7  283.   6.46e7
3 K04000~ ENGL~ Country  1.51e5  5.97e7  395.   5.94e7  394.   5.91e7
4 E92000~ ENGL~ Country  1.30e5  5.66e7  434.   5.63e7  432.   5.60e7
5 E12000~ NORT~ Region   8.58e3  2.68e6  312.   2.67e6  311.   2.66e6
6 E06000~ Coun~ Unitar~  2.23e3  5.33e5  240.   5.30e5  238.   5.27e5
```

```

7 E06000~ Darl~ Unitar~ 1.97e2 1.07e5 544. 1.07e5 541. 1.07e5 540.
8 E06000~ Hart~ Unitar~ 9.37e1 9.38e4 1001. 9.37e4 999. 9.32e4 995.
9 E06000~ Midd~ Unitar~ 5.39e1 1.41e5 2622. 1.41e5 2616. 1.41e5 2608.
10 E06000~ Nort~ Unitar~ 5.02e3 3.24e5 64.5 3.22e5 64.2 3.20e5 63.8
# ... with 410 more rows, 34 more variables:
#   `Estimated Population mid-2017` <dbl>, `2017 people per sq. km` <dbl>,
#   `Estimated Population mid-2016` <dbl>, `2016 people per sq. km` <dbl>,
#   `Estimated Population mid-2015` <dbl>, `2015 people per sq. km` <dbl>,
#   `Estimated Population mid-2014` <dbl>, `2014 people per sq. km` <dbl>,
#   `Estimated Population mid-2013` <dbl>, `2013 people per sq. km` <dbl>,
#   `Estimated Population mid-2012` <dbl>, `2012 people per sq. km` <dbl>, ...

```

For the assignment, you should only work with smaller sub-national areas. Filter out country and regional area. You should address the following questions:

1. Use sequence analysis to identify representative trajectories of population change and discuss the type of trajectories identified in London Boroughs.
2. Use individual sequence plot to identify distinctive features in the resulting trajectories.

For the analysis, aim to focus on the area of London so you can link your narrative to the rest of analyses you will be conducting.

Ensure you justify the number of optimal clusters you will use in your analysis and provide a brief description of the trajectories identified. Describe how they are unique.

# 5 Network Analysis

Network Analysis is an interdisciplinary analytical framework which studies the structure, behavior, and dynamics of the connections between different elements. Specific applications range from the analysis of social links between people to the analysis of transportation and migration links between places.

In particular, network analysis has proven to be highly effective for gaining insights about the behavior of populations, and the spatial distribution of people and resources (Prieto Curiel, Cabrera-Arnau, and Bishop 2022). By examining the networks of human interactions, researchers can gain a better understanding of how social, economic, and cultural factors influence individual behavior, and how patterns of migration and communication can shape the development of communities, regions and cities (Cabrera-Arnau et al. 2022). By the end of this session, you should be able to describe some of the most fundamental concepts and tools for network analysis. You should also be able to understand the potential of this approach to revolutionise the way in which we study populations.

The chapter is based on the following references:

- Network analysis with R and igraph: NetSci X Tutorial (Ognyanova 2016).
- [igraph](#) - The network analysis package.

The book Networks by Newman (2018) is an excellent resource to learn more about network analysis, which covers the basics but also the more advanced concepts and methods.

## 5.1 Dependencies

To run the code in the rest of this workbook, we will need to load the following R packages:

```
#Support for simple features, a standardised way to encode spatial vector data
library(sf)
#Data manipulation
library(dplyr)
# An R package for network manipulation and analysis
library(igraph)
# Provides a number of useful functions for working with character strings in R
```

```
library(stringr)
```

## 5.2 Data

### 5.2.1 The US Census dataset

In this Chapter we will be looking at data provided by [US Census Bureau](#). In particular, we have prepared the file metro\_to\_metro\_2015\_2019\_US\_migration.csv, which contains migration data between the US Metropolitan Statistical Areas (MSAs) in two consecutive years. The data is based on the 2015-2019 American Community Survey (ACS) estimates. For more information on the methodology for the data collection process, visit this [link](#).

### 5.2.2 Import the data

Before importing the data, ensure to set the path to the directory where you stored it. Please modify the following line of code as needed.

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

Let us do a few changes in the names of the fields of the dataset so we can later manipulate them more easily.

```
#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  
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)
```

Each row corresponds to an origin-destination pair, so we can obtain the total number of reported migratory movements with the following command:

```
nrow(df)
```

```
[1] 52930
```

## 5.3 Creating networks

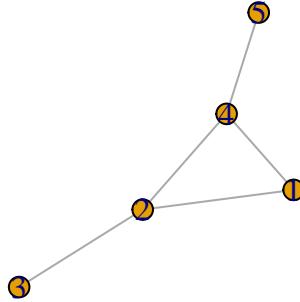
Before we start to analyse the data introduced in Section 5.2, let us first take a step back to consider the main object of study of this Chapter: the so-called networks. In the most general sense, a **network** (also known as a graph) is a structure formed by a set of objects which may have some connections 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.

```
# Create an undirected network with 5 nodes and 5 edges
# 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
g1 <- graph( edges=c(1,2, 1,4, 2,3, 2,4, 4,5), n=5, directed=F )

# A simple plot of the network allows us to visualise it
plot(g1)
```



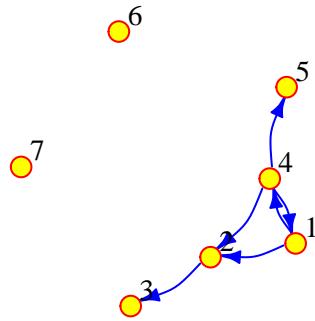
If the connections between the nodes of a network are non-reciprocal, the network is called directed. For example, this could correspond to a situation where there are people moving from city 1 to city 2, but nobody moving from city 2 to city 1. Note that in the code below we have not only added directions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

```

# Creates a directed network with 7 nodes and 6 edges
# Note that we now have edge 1,4 and edge 4,1 and that 2 of the nodes are isolated
g2 <- graph( edges=c(1,2, 1,4, 2,3, 4,1, 4,2, 4,5), n=7, directed=T )

# A simple plot of the network with a few extra features
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

```

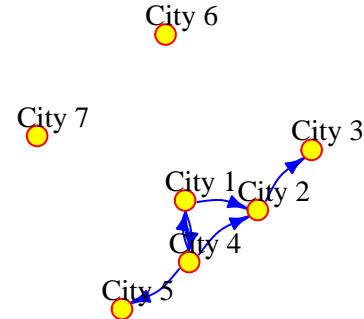


The network can also be defined as a list containing pairs of named nodes. Then, it is not necessary to specify the number of nodes but the isolated nodes have to be included. The following code generates a network which is equivalent to the one above.

```

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 7dd5643:
[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 7dd5643 (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\_Prevous\_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
```

```

# Creates the df_edges data frame with data from df and renames the columns as "origin", "destination" and "weight"
df_edges <- data.frame(df$MSA_Previous_Name_State, df$MSA_Current_Name_State, df$Movers_Mean_Distance)
  rename(origin = df.MSA_Previous_Name_State, destination = df.MSA_Current_Name_State, weight = df.Movers_Mean_Distance)

#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.

```

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.

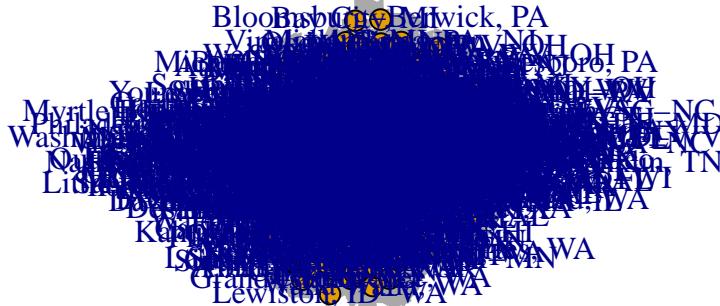
```

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...

```
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, Washington. 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.

```
# Filter the original data frame
df_sub <- df %>% filter(grepl('WA', MSA_Current_State)) %>% filter(grepl('WA', MSA_Previous
```

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  $df\_sub$ .

```

# Create a new data frame containing the columns for the origin, destination, and weight
# Rename the columns to origin, destination, and weight respectively
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 = df_sub.Mover)

# Split long names into several lines for better visualization in the resulting graph
df_sub_edges$origin <- gsub("-", "-\n", df_sub_edges$origin)
df_sub_edges$destination <- gsub("-", "-\n", df_sub_edges$destination)

# Convert the weight column to numeric, removing any commas that may be present
df_sub_edges$weight <- as.numeric(gsub(",","",df_sub_edges$weight))

# Create a data frame of unique origins by selecting distinct values of MSA_Previous_Name
# Rename the resulting column to "name"
df_sub_unique_origins <- df_sub %>%
  distinct(MSA_Previous_Name) %>%
  rename(name = MSA_Previous_Name)

# Create a data frame of unique destinations by selecting distinct values of MSA_Current_Name
# Rename the resulting column to "name"
df_sub_unique_destinations <- df_sub %>%
  distinct(MSA_Current_Name) %>%
  rename(name = MSA_Current_Name)

# Merge the unique origins and unique destinations data frames into one data frame of nodes
# Match the rows based on the "name" column
df_sub_nodes <- full_join(df_sub_unique_origins, df_sub_unique_destinations, by = "name")

# Split long names into several lines for better visualization in the resulting graph
df_sub_nodes$name <- gsub("-", "-\n", df_sub_nodes$name)

# Create a graph object from the edges and nodes data frames
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

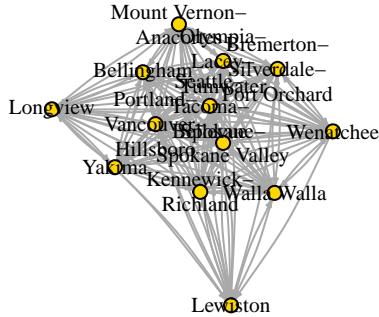
Let us start by generating the most basic visualisation of `g_sub`.

```
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  $0.4 \times (V(gsub)\$inflow)^{0.4}$ , 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.

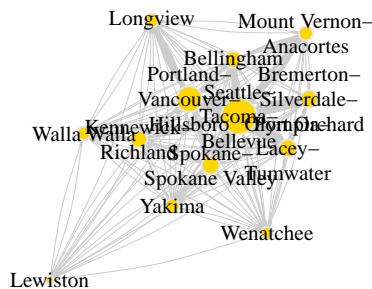
```
# Set node size based on inflow of migrants:  
V(g_sub)$size <- 0.4*(V(g_sub)$inflow)^0.4  
# Set edge width based on weight:  
E(g_sub)$width <- E(g_sub)$weight/1200
```

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`.

```

# Import core-based statistical areas https://www.census.gov/geographies/mapping-files/tim
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
```

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

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:

```
# Create a new column with the name of the state taken from the last two characters of ent
msa_us$NAME_ONLY <- gsub("..*\"", "", msa_us$NAME)

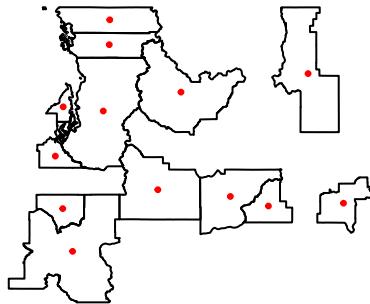
# Long names of MSAs are split into lines for visualisation purposes
msa_us$NAME_ONLY <- gsub("-", "-\n", msa_us$NAME_ONLY)

# Create a new column with the name of the state taken from the last two characters of ent
msa_us$STATE <- substr(msa_us$NAME, nchar(msa_us$NAME)-1, nchar(msa_us$NAME))

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

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)
```

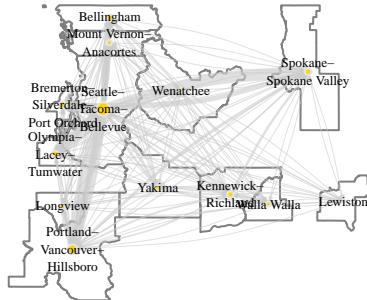


However, we still need to link this data to the network data that we obtained before. In order to incorporate the geographic information to the nodes of the migration subnetwork, we can join data from two data frames: *msa\_sub*, which contains the geographic data, and *df\_sub\_nodes*, which contains 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 centroids
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, 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
```

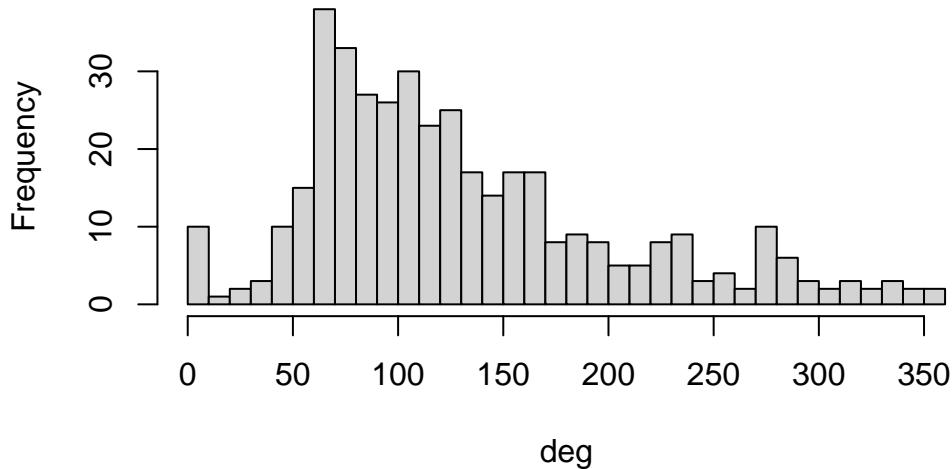
From this result, we conclude that about 62% of the pairs of nodes that are connected have edges in both directions.

### 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-deg  
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")
```

## 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")
```

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

A **path** in a network between node  $A$  and node  $B$  is a sequence of edges which joins a sequence of distinct nodes, starting at node  $A$  and terminating at node  $B$ . In a **directed path** there is an added restriction: the edges must be all directed in the same direction.

The **length of a path** between nodes  $A$  and  $B$  is normally defined as the number of edges that form the path. The **shortest path** is the minimum number of edges that need to be traversed to travel from  $A$  to  $B$ .

The length of a path can also be defined in other ways. For example, if the edges are weighted, it can be defined as the sum of the weights of the edges that form the path.

In R, we can use the function **shortest\_paths()** to find the shortest path between a given pair of nodes and its length. For example, below we can see that the shortest path between the MSA containing New York and the MSA containing Los Angeles is one. This is not surprising since we would expect an edge connecting these two MSAs representing the fact that there is people migrating from one to the other in two consecutive years.

```
shortest_paths(g_US,
from = V(g_US)$name=="New York-Newark-Jersey City, NY-NJ-PA",
to = V(g_US)$name=="Los Angeles-Long Beach-Anaheim, CA",
weights=NA, #If weights=NULL and the graph has a weight edge attribute, then the weight at
output = "both") # outputs both path nodes and edges
```

```
$vpath
$vpath[[1]]
+ 2/402 vertices, named, from 534cb6f:
[1] New York-Newark-Jersey City, NY-NJ-PA Los Angeles-Long Beach-Anaheim, CA
```

```
$epath
$epath[[1]]
+ 1/52930 edge from 534cb6f (vertex names):
[1] New York-Newark-Jersey City, NY-NJ-PA->Los Angeles-Long Beach-Anaheim, CA
```

```
$predecessors  
NULL
```

```
$inbound_edges  
NULL
```

Of all shortest paths in a network, the length of the longest one is defined as the **diameter** of the network. In this case, the diameter is 3 meaning that the longest of all shortest paths in *g\_US* has 3 edges.

```
diameter(g_US, directed=TRUE, weights=NA)
```

```
[1] 3
```

The mean distance is defined as the average length of all shortest paths in the network. The mean distance will always be smaller or equal than the diameter.

```
mean_distance(g_US, directed=TRUE, weights=NA)
```

```
[1] 1.663335
```

### 5.6.5 Centrality

Centrality metrics assign scores to nodes (and sometimes also edges) according to their position within a network. These metrics can be used to identify the most influential nodes.

We have already explored some concepts which can be regarded as centrality metrics, for example, the **degree** of a node or the weighted degree of a node, also known as the **strength of a node**, which is the sum of edge weights that link to adjacent nodes or, in other words, the in-flow or out-flow associated with each node. As we can see from the code below, many nodes in *g\_US* have an in-flow of less than 100 immigrants. Note that we have set mode to c("in") for inflow and we have set the weights parameter to NULL since we want to know the sum of weights for the incoming edges and not just the total number of incoming edges.

```
# Compute strength of the nodes belonging to graph g_US, in this case the in-flow  
strength_US <- strength(g_US, #The input graph  
vids = V(g_US), # The vertices for which the strength will be calculated.  
mode = c("in"), # "in" for in-degree  
loops = FALSE, #whether the loop edges are also counted
```

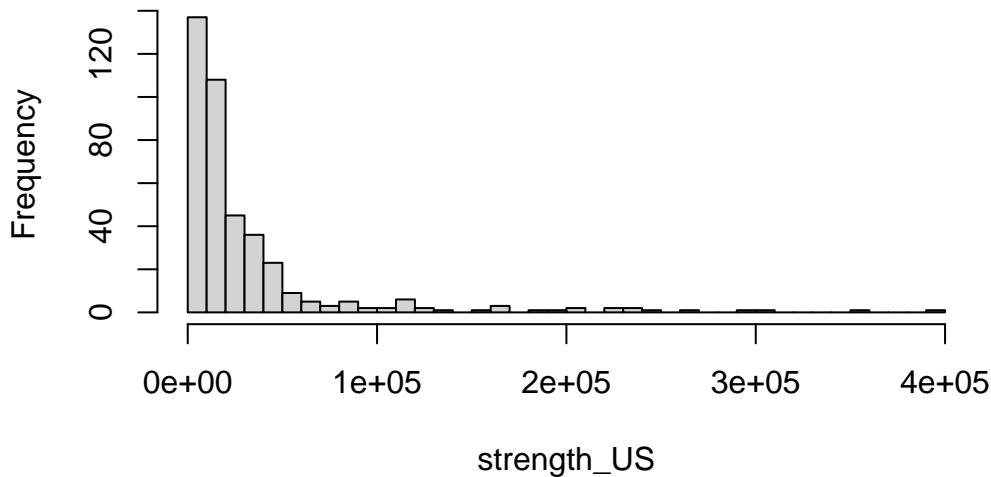
```

    weights = NULL #If the graph has a weight edge attribute, then this is used by default
  )

#Produce histogram of the frequency of nodes with a certain strength
hist(strength_US, breaks = 50, main="Histogram of node strength")

```

## Histogram of node strength



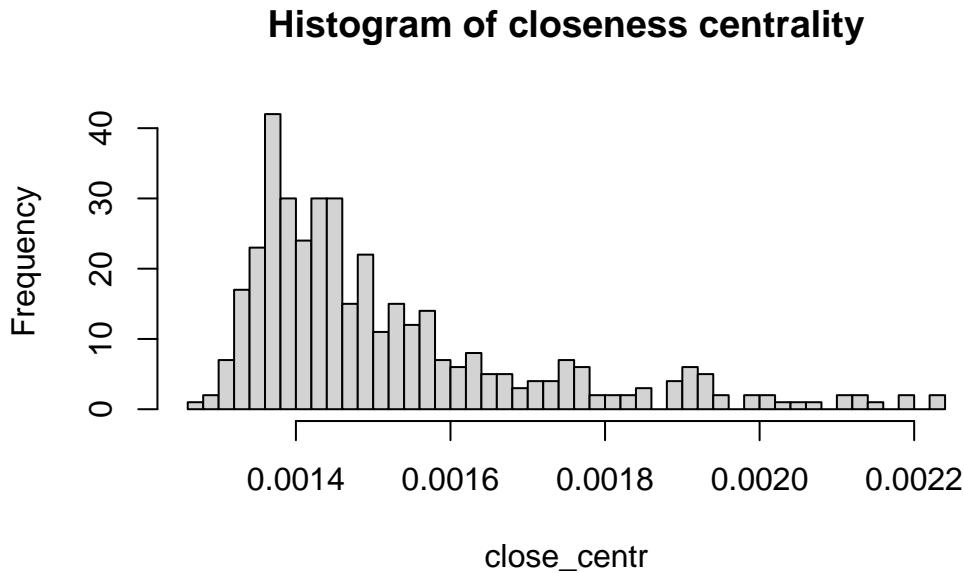
We can check which is the city with the maximum strength:

```
V(g_US)$name[strength(g_US, vids = V(g_US), mode = c("in"), loops = FALSE, weights = NULL)]
```

```
[1] "New York-Newark-Jersey City, NY-NJ-PA"
```

We will look at another two important centrality metrics that are based on the structure of the network. Firstly, **closeness centrality** which is a measure of the length of the shortest path between a node and all the other nodes. For a given node, it is computed as the inverse of the average shortest paths between that node and every other node in the network. So, if a node has closeness centrality close to 1, it means that on average, it is very close to the other nodes in the network. A closeness centrality of exactly 0 corresponds to an isolated node.

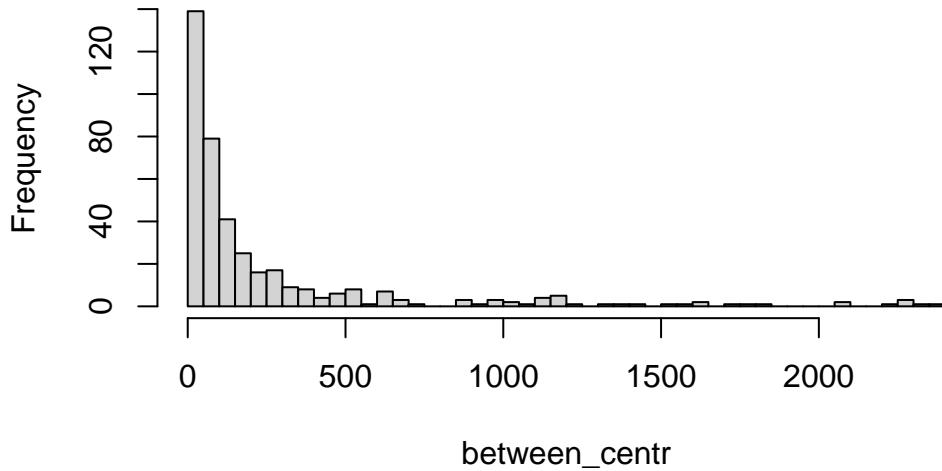
```
close_centr <- closeness(g_US, mode="in", weights=NA) #using unweighted edges  
hist(close_centr, breaks = 50, main="Histogram of closeness centrality")
```



The other metric is known as **betweenness centrality**. For a given node, it is a measure of the number of shortest paths that go through that node. Therefore, nodes with high values of betweenness centrality are those that play a very important role in the connectivity of the network. Betweenness can also be computed for edges.

```
between_centr <- betweenness(g_US, v = V(g_US), directed = TRUE, weights = NA)  
hist(between_centr, breaks = 50, main="Histogram of betweenness centrality")
```

## Histogram of betweenness centrality

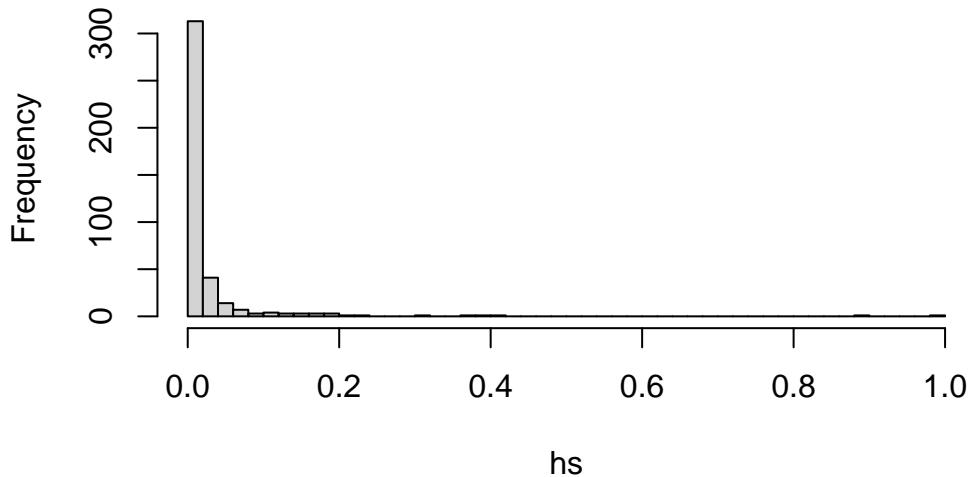


### 5.6.6 Hubs and authorities

We call hubs or authorities those nodes with a higher-than-average degree. Normally, the name hub is reserved to nodes with high out-degree whereas authority is reserved to nodes with high in-degree. An algorithm to detect hubs and authorities was developed by Jon Kleinberg, although it was initially used to examine web pages. Like we did for other network metrics, we can compute the hub score and then plot a histogram to see how this metric is distributed across the nodes of the network.

```
hs <- hub_score(g_US, weights=NULL)$vector #In this case, we use the weighted edges  
hist(hs, breaks = 50, main="Histogram of hub score")
```

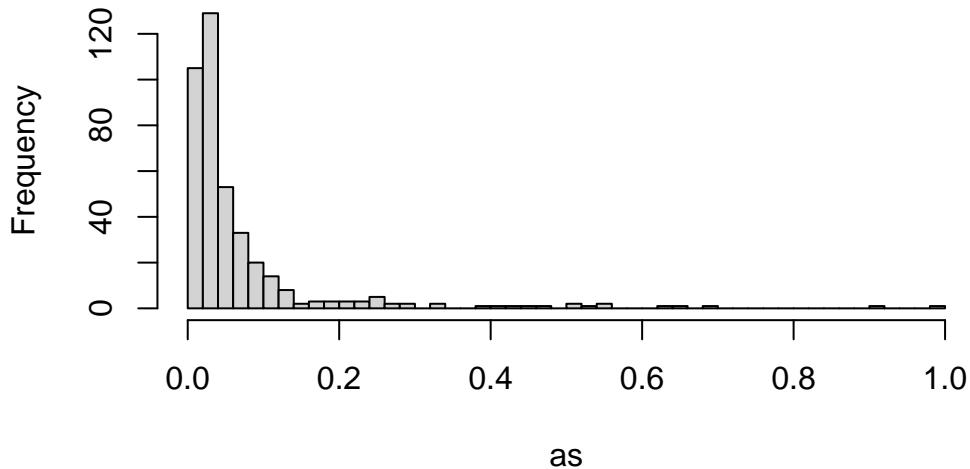
### Histogram of hub score



Similarly, we can explore the authority score for each node:

```
as <- authority_score(g_US, weights=NULL)$vector  
hist(as, breaks = 50, main="Histogram of authority score")
```

## Histogram of authority score



## 5.7 Questions

In this set of questions, we will use internal migration data corresponding to the London boroughs. The original data can be found on the UK Office for National Statistics [website](#). The data set that we use below corresponds for the year ending in June 2019 and has already been cleaned for you. You can import it with the following line of code:

```
df <- read.csv("./data/networks/LA_to_LA_2019_London_clean.csv")
```

Essay questions:

1. Create a network with igraph that represents the migratory movements between the London boroughs in the year ending in June 2019. The nodes of this network will represent the different boroughs and the edges will represent the migration flows between them. Make sure the network object is directed.
2. Produce histograms for the unweighted in-degree and out-degree of the nodes in the network. Produce histograms for the weighted in-degree and out-degree (also known as strength of inflows and outflows) as well. Combine all four histograms in one plot. Additionally, compute the edge density of the network (with no loops). Comment on all your observations. Finally, based on network metrics, what is the code for the London

borough with the largest incoming population as well as the London borough with the largest outgoing population?

3. BONUS QUESTION. This question **will not be assessed** but it is an excellent exercise for those of you who are keen on networks. Create a visualisation of a migration network showing the flows between different boroughs. You can get as creative as you like. You may use igraph or other tools that have not been discussed in this workbook but that you may want to explore by yourself.

You are most welcome to show your figure to one of the lecturers for an opportunity to get feedback. However, since it is not assessed, you are **not supposed to include it in the essay**.

For your visualisation, you may want to use geographical data for the London boroughs. To access it, you should run the code below. It loads the necessary data for the geographical boundaries of the Local Authority Districts (LADs) in England and Wales and then, it filters the dataset so only the London boroughs remain, i.e. those LADs starting with E09. The geographical boundaries can be downloaded from the ONS Open Geography Portal [website](#) but we have also included a copy of the dataset in the GitHub repository associated with this module.

```
# Import boundaries for local authorities in England and Wales
LA_UK <- st_read("./data/networks/Local_Authority_Districts_(December_2022)_Boundaries_UK_"

Reading layer `LAD_DEC_2022_UK_BFC' from data source
`/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202223/r4ps/data/networks/Local_Authority_Districts_(December_2022)_Boundaries_UK_BFC.shp'
using driver `ESRI Shapefile'
Simple feature collection with 374 features and 10 fields
Geometry type: MULTIPOLYGON
Dimension:      XY
Bounding box:  xmin: -116.1928 ymin: 5336.966 xmax: 655653.8 ymax: 1220302
Projected CRS: OSGB36 / British National Grid

# Filter the original data frame to obtain only London boroughs
LND_boroughs <- LA_UK %>% filter(grepl('E09', LAD22CD))

plot(st_geometry(LND_boroughs))
```



# 6 Sentiment Analysis

This chapter illustrates the use of sentiment analysis and social media data to measure and monitor public opinion sentiment about migration on Twitter. Sentiment analysis, also known as opinion mining or emotion artificial intelligence, refers to the use of natural language processing to systematically identify, measure and analyse emotional states and subjective information. It computationally identifies the polarity of text, that is, whether the underpinning semantics of an opinion is positive, negative or neutral. It allows deriving quantitative scores to identify the attitude or position on the distribution of negative or positive terms in a given piece of text.

This chapters focuses on migration sentiment. Immigration has consistently been identified as one of the most divisive social issues globally (European Commision 2019). Immigration sentiment shapes migration policy formulation and political outcomes. Anti-immigration sentiment has spurred attention towards more restrictive migration policies, and has been linked to an increasing prominence of right-wing affiliation, particularly in Western European countries and the United States (e.g. Bail et al. 2018). Immigration sentiment also influences the capacity of migrants to integrate into receiving communities. Acts of discrimination, intolerance and xenophobia can impair immigrants' ability to secure employment, housing and achieve a sense of belonging in local communities, contributing to more polarised societies (Cheong et al. 2007).

Traditionally, data on public perception on attitudes towards immigration have been collected through qualitative sources, namely ethnographies, interviews and surveys. Yet, qualitative methods rely on small samples and normally suffer from sample bias (Rowe 2021a). Similarly, while surveys can provide a reliable national representation, they are expensive, infrequent, offer low population coverage, lack statistically validity at fine geographical scales and become available with a lag of one or two years after they have been collected. Social media, particularly microblogging, has been identified as a key source of data to overcome these limitations. Social media offers a dynamic and open space which provides a unique window to better understand public opinion about immigration. As stated previously, this chapter aims to illustrate how Twitter data and sentiment analysis can be used to measure sentiment about migration.

The chapter is based on the following references:

- R for Data Science: [Working with strings \(Chapter 14\)](#).
- Text Mining with R: [The tidy text format](#).
- Hutto and Gilbert (2014) on a widely used sentiment analysis algorithm for Twitter data.

- Rowe, Mahony, Graells-Garrido, et al. (2021), Rowe, Mahony, Sievers, et al. (2021) illustrate the use of Twitter data and sentiment analysis to monitor migration sentiment during the COVID-19 pandemic.

## 6.1 Dependencies

```
library(tidytext) # text data tidy approach
library(tm) # creating data corpus
library(SnowballC) # stemming text
library(tidyverse) # data manipulation

# sentiment analysis
library(vader)

# download twitter data (no used)
library(rtweet)

# design plots
library(patchwork)
```

## 6.2 Data

We will use a sample of Twitter data on public opinion about migration originated in the United States during December 1st 2019 and May 1st 2020. They data were collected by Rowe, Mahony, Graells-Garrido, et al. (2021) to analyse changes in public opinion related to migration during the early stages of the COVID-19 pandemic. During this period, a rising number of anti-immigration sentiment incidents were reported across the world (“Stop the Coronavirus Stigma Now” 2020). Acts and displays of intolerance, discrimination, racism, xenophobia and violent extremism emerged linking individuals of Asian descent and appearance to COVID-19 Coates (2020). In the United States, President Donald Trump repeatedly used the terms “Chinese Virus,” “China Virus,” and “Fung Flu” in reference to COVID-19. Fear mongering and racial stereotyping spread on social media and rapidly spilled onto the streets (Cowper 2020). In the United Kingdom, the government reported a 21% increase in hate crime incidents against Asian communities between January and March, and Chinese businesses reported a notorious reduction in footfall during Chinese celebrations (Home Affairs Committee 2020).

Data were collected via an application programming interface (API) using random sampling strategy. A set of key search terms were defined, including words, Twitter accounts and hashtags to collect the data. These search terms were developed in collaboration with the

United Nations' International Organization for Migration. See Rowe, Mahony, Graells-Garrido, et al. (2021) for details on the search strategy. For this chapter, Twitter users' handles have been anonymised.

```
tweets_df <- readRDS("./data/sentiment-analysis/usa_tweets_01122019_01052020.rds")
glimpse(tweets_df)
```

```
Rows: 47,950
Columns: 4
$ id          <dbl> 176661, 176662, 176663, 176664, 176665, 176666, 176667, 176-
$ created_at <dttm> 2020-03-30 23:02:04, 2020-03-30 23:02:18, 2020-03-30 23:03-
$ status_id   <dbl> 1.244762e+18, 1.244762e+18, 1.244762e+18, 1.244763e+18, 1.2-
$ text        <chr> "@anonymous @anonymous @anonymous @anonymous @an-
```

### 6.2.1 Text data structures

General data frame object has a specific structure: Each variable is a column Each observation is a row Each type of observational unit is a table

Different approaches to storing and manipulating text data exist:

String: Text can be stored as strings, i.e., character vectors, within R, and often text data is first read into memory in this form.

Corpus: These types of objects typically contain raw strings annotated with additional meta-data and details.

Document-term matrix: This is a sparse matrix describing a collection (i.e., a corpus) of documents with one row for each document and one column for each term. The value in the matrix is typically word count.

Introduce TidyText with an example: We thus define the tidy text format as being a table with one-token-per-row. A token is a meaningful unit of text, such as a word, that we are interested in using for analysis, and tokenisation is the process of splitting text into tokens. This one-token-per-row structure is in contrast to the ways text is often stored in current analyses, perhaps as strings or in a document-term matrix. For tidy text mining, the token that is stored in each row is most often a single word, but can also be an n-gram, sentence, or paragraph. In the tidytext package, we provide functionality to tokenize by commonly used units of text like these and convert to a one-term-per-row format. Tidy data sets allow manipulation with a standard set of “tidy” tools,

```
# converting to a tidytext data representation
tidy_tweets_df <- tweets_df %>%
```

```
select(created_at, text) %>%  
unnest_tokens("word", text)
```

In numeric data analysis, we normally look at the distribution of variables to gain a first understanding of the data. In text analysis, we explore the distribution of words. Typically we analyse the top words.

```
# ranking words  
tidy_tweets_df %>%  
  count(word) %>%  
  arrange(desc(n))  
  
# A tibble: 50,583 x 2  
  word           n  
  <chr>        <int>  
1 anonymous    102087  
2 the          47883  
3 to           36734  
4 http          29758  
5 url_removed  28382  
6 and          24126  
7 of            23356  
8 in            19748  
9 a             18451  
10 is           16069  
# ... with 50,573 more rows
```

### 6.2.2 Basic text data principles

Working with text data is complex. There are various important concepts and procedures that we need to introduce to get you familiar with, before we can get you rolling with text data mining. In this section, we introduce key concepts using example to illustrate the main ideas and basic code.

#### Character encoding

Character encoding is the numeric representation of graphical characters that we use to represent human language. Character encoding, such as UTF-8, ASCII and -ISO-8859-1 enables characters to be stored, transmitted and transformed using digital computers. For example, we use the English alphabet and understand differences between lower, upper case letters, numerals and punctuation. Computers encode and understand these characters as binary numeric combinations. There is no unique system of character representation.

Various systems exist and vary according to the type of information, when it was stored and geographic context. Additionally, different character encoding representations are used with varying levels of popularity according to the operating system, language and software, and this level of popularity changes over time. Currently UTF-8 is one of the most popular character encoding systems used on the web according to [Google](#).

Sometimes we may need to standardise text data before they can be combined based on different character encoding systems. Generally R recognises and read different character encoding representations. But, if you notice an error of invalid string or an unusual character, this may mean you are using a dataset based on a character encoding representation which has not been globally integrated in R. This tends to occur with characters in various languages and emojis.

There is not quick way to standardise two different encoding systems, to our knowledge. Two handy functions in R that would help you with this task is the `iconv` and `encoding`. Both functions require you to know the character encoding representation of the data.

### Regular expressions

Regular expressions are patterns that occur in a group of strings. Often when you work with text data, you are likely to find unusual character expressions, particularly if you are working with data scrapped from a website. Strings such as `tab` and `return` are normally represented by `\t` and `\r` so you may want to remove these expressions. To deal with regular patterns, we can use the `grep` base R library. In our dataset, we do have various occurrences of the expression `&gt;` which stands for `>` - see example below. Let's assume we want to remove this expression from our tweet sample. We first could use `grepl` from the `grep` library to identify tweets containing this expression. The `grepl` function asks if the first expression before the comma is present in the data object after the comma, and returns Boolean result i.e. `TRUE` or `FALSE`.

#### Note

You can also use the function `str_detect` from the `stringr` package to identify regular patterns or expressions in text

```
grepl("&gt;", tweets_df$text[1:50])
```

```
[1] FALSE  
[13] FALSE  
[25] FALSE  
[37] FALSE  
[49] FALSE FALSE
```

Let's see one of those tweets:

```
tweets_df$text[30]
```

```
[1] "@anonymous @anonymous @anonymous @anonymous @anonymous @anonymous @anonymous @anonymous
```

We may now want to remove the regular expression `&gt;`. We can do this by using `gsub` from the `grep` library. In `gsub`, you add the pattern you want to replace (i.e. `&gt;`), followed by the expression you want to use to replace this pattern with (nothing), and the data object (the tweet:`tweets_df$text[30]`).

```
gsub("&gt;", "", tweets_df$text[30])
```

```
[1] "@anonymous @anonymous @anonymous @anonymous @anonymous @anonymous @anonymous @anonymous
```

`grep` functions can also be very helpful in identifying a word and prefix in a string of words. We recommend consulting the [Cheat Sheet](#) for basic regular expressions in R put together by Ian Kopacka, to get familiar with the various character expressions and functions.

### Unit of analysis

A key component in text data mining is the unit of analysis. We could focus our analysis on single words, individual sentences, paragraphs, sections, chapters or a larger corpus of text. The concept of here is relevant. The process of splitting text into units is known as *tokenisation*. Tokens are the resulting units of analysis. In the context of text data mining, *n-grams* is a popular tokenisation concept. This refers to a sequence of words of length n. A unigram is one word (e.g. migration). A bigram is a sequence of two words (migration can). A trigram is a sequence of three words (migration can have) and so on. n-grams can be useful when you want to capture the meaning of a sequence of words; for example, identifying “The United Kingdom” in a sequence of words. In R, we can use `tidytext` to organise the data according to n-grams.

#### Note

Note that the size of the chunk of text that we use to add up unigram sentiment scores can have an effect on an analysis. A text the size of many paragraphs can often have positive and negative sentiment averaged out to about zero, while sentence-sized or paragraph-sized text often works better. Similarly, small sections of text may not contain enough words to accurately estimate sentiment, while sentiment in very large sections may be difficult to identify.

```

tweets_df[1:50,] %>%
  select(created_at, text) %>%
  unnest_tokens(ngram, text, token = "ngrams", n = 2)

# A tibble: 1,369 x 2
  created_at      ngram
  <dttm>        <chr>
1 2020-03-30 23:02:04 anonymous anonymous
2 2020-03-30 23:02:04 anonymous anonymous
3 2020-03-30 23:02:04 anonymous anonymous
4 2020-03-30 23:02:04 anonymous anonymous
5 2020-03-30 23:02:04 anonymous anonymous
6 2020-03-30 23:02:04 anonymous anonymous
7 2020-03-30 23:02:04 anonymous no
8 2020-03-30 23:02:04 no one
9 2020-03-30 23:02:04 one is
10 2020-03-30 23:02:04 is above
# ... with 1,359 more rows

```

### 💡 Tip

#### Task

Try changing `n` to 4. What changes do you observe?

## Text pre-processing

We also need to think about the words we want to include in our analysis. Normally we focus on a selection of words conveying a particular conceptual representation. Some words may not convey much meaning, enrich our analysis or may distort the meanings we want to capture. So carefully thinking of the words we want to include in our analysis is important. Below we go through key character concepts which are often considered in text data mining and natural language processing. These concepts imply the removal of certain characters such as stop words, punctuation and numbers. The need to remove these characters will vary on the aim of the study, context and algorithm used to analyse the data.

### *Stop words*

Stop words are commonly used words in a language. They tend to comprise articles, prepositions, pronouns and conjunctions. Examples of stop words in English are “a”, “the”, “is” and “are”. Stop words are essential to communicate in every day language. Yet, in the text data mining and NLP, stop words are often removed as they are considered to carry limited useful information. Stop words differ across languages and different lists of words exist to remove stop words from analysis. We use the `tidytext` approach to remove stop words by running:

```
# remove stop words
data("stop_words") # call stop word list
tidy_tweets_df <- tidy_tweets_df %>%
  anti_join(stop_words)
```

Joining, by = "word"

Let's see what are the top words now:

```
tidy_tweets_df[1:10,] %>%
  count(word) %>%
  arrange(desc(n))
```

```
# A tibble: 4 x 2
  word      n
  <chr>    <int>
1 anonymous     7
2 law           1
3 people        1
4 threatening   1
```

We can see words, such as “t.co” and “https” which may not add much to our analysis and we may consider to remove them using `grep` functions. We will illustrate this below.

#### *Punctuation*

We may also want to remove punctuation. Again, while punctuation may be very important in written language to communicate and understand the meaning of text. Punctuation by itself does not convey helpful meaning in the context of text analysis as we often take the text out of sequential order. Punctuation removal is another reason to prefer `tidytext` as punctuation marks are removed automatically.

#### *Numbers*

We may want to remove numbers. Sometimes numbers, such as 9/11 or 2016 may provide very relevant meaning in the terrorist attacks in the US context or Brexit referendum in the UK. However, they generally do not add much meaning. We can use `grep` to remove numbers. The "\b\d+\b" text tells R to remove all numeric digits. `d` stands for digits. The `-` sign tells `grep` to exclude these digits.

```
# remove numbers
tidy_tweets_df <- tidy_tweets_df[ -grep("\b\d+\b",
```

```
tidy_tweets_df$word),]
```

### *Word case*

An additional element to consider is word case. Often all text is forced into lower case in quantitative text analysis as we do not want words starting with an upper case word such as “Migration” to be counted as a different word than “migration”. In this example, the difference between using lower or upper case words may not matter. Semantically, on other occasions (e.g. in a sentiment analysis context), this distinction may make all the difference. Consider “HAPPY” or “happy”. The former may emphasise that a person is much happier than in the latter case and we may want to capture the intensity of this emotion in our analysis. In such cases, we may be better off preserving the original text.

The `unnest_tokens` in the `tidytext` package automatically forces all words into lower case. To preserve upper case words, you will need to change the default options for `to_lower` to `FALSE`.

```
tweets_df[1:10,] %>%
  select(created_at, text) %>%
  unnest_tokens("word",
    text,
    to_lower = FALSE) # preserve upper case

# A tibble: 243 x 2
  created_at      word
  <dttm>        <chr>
1 2020-03-30 23:02:04 anonymous
2 2020-03-30 23:02:04 anonymous
3 2020-03-30 23:02:04 anonymous
4 2020-03-30 23:02:04 anonymous
5 2020-03-30 23:02:04 anonymous
6 2020-03-30 23:02:04 anonymous
7 2020-03-30 23:02:04 anonymous
8 2020-03-30 23:02:04 No
9 2020-03-30 23:02:04 one
10 2020-03-30 23:02:04 is
# ... with 233 more rows
```

### *White spaces*

White spaces can also be concern. Often white spaces may also be considered as words. In `tidytext` language, white spaces can be removed using the `gsub` function identifying white spaces with `s+`.

```

gsub("\\s+",
  "",
  tidy_tweets_df$word[1:20])

```

```

[1] "anonymous"    "anonymous"    "anonymous"    "anonymous"    "anonymous"
[6] "anonymous"    "anonymous"    "law"          "people"       "threatening"
[11] "republican"   "government"   "president"   "arrested"     "http"
[16] "url_removed" "anonymous"   "approve"      "ice"         "delivers"

```

### *Stemming*

A common step in text-preprocessing is stemming. Stemming is the processing of lowering inflection in words to their root forms. For example, the stem of the word “monitoring” is “monitor”. This step aids in the pre-processing of text, words, and documents for text normalisation. Stemming is common practice because we do not want the words, such as “monitoring” and “monitor” to convey different meanings to algorithms, as such topic modelling algorithms, that we use to extract latent themes from unstructured texts.

For stemming words, we can use the function `wordStem` from the `SnowballC` package.

```

tidy_tweets_df[1:20,] %>%
  mutate_at("word",
    funs(wordStem(.),
      language="en")))

```

```

Warning: `funs()` was deprecated in dplyr 0.8.0.
i Please use a list of either functions or lambdas:

```

```

# Simple named list: list(mean = mean, median = median)

# Auto named with `tibble::lst()`: tibble::lst(mean, median)

# Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))

# A tibble: 20 x 2
  created_at        word
  <dttm>           <chr>
1 2020-03-30 23:02:04 anonym
2 2020-03-30 23:02:04 anonym
3 2020-03-30 23:02:04 anonym
4 2020-03-30 23:02:04 anonym

```

```

5 2020-03-30 23:02:04 anonym
6 2020-03-30 23:02:04 anonym
7 2020-03-30 23:02:04 anonym
8 2020-03-30 23:02:04 law
9 2020-03-30 23:02:04 peopl
10 2020-03-30 23:02:04 threaten
11 2020-03-30 23:02:04 republican
12 2020-03-30 23:02:04 govern
13 2020-03-30 23:02:04 presid
14 2020-03-30 23:02:04 arrest
15 2020-03-30 23:02:04 http
16 2020-03-30 23:02:04 url_remov
17 2020-03-30 23:02:18 anonym
18 2020-03-30 23:02:18 approv
19 2020-03-30 23:02:18 ice
20 2020-03-30 23:02:18 deliv

```

#### *Coding characters*

Typically we also want to remove coding characters, including links to websites. Words such as “https”, “t.co” and “amp” are common in webscrapped or social media text. The strings “https” and “t.co” appear as the top two more frequent words in our data. Generally such words do not convey relevant meaning for the analysis so they are removed. To do this, we can use `grep`.

```

tidy_tweets_df[!grep("https|t.co|amp",
tidy_tweets_df$word),] %>%
  count(word) %>%
  arrange(desc(n))

```

```

# A tibble: 47,991 x 2
  word          n
  <chr>     <int>
1 anonymous    102087
2 http         29758
3 url_removed  28382
4 immigration   9034
5 covid19       6115
6 coronavirus    5190
7 china          5149
8 people          4991
9 chinesevirus    4963

```

```
10 chinavirus      4951
# ... with 47,981 more rows
```

We could also use `str_detect` from the `stringr` package.

```
library(stringr)
library(dplyr)

coding_words <- c("https|t.co|amp")

tidy_tweets_df <- tidy_tweets_df %>%
  filter(!str_detect(word, coding_words))

tidy_tweets_df %>%
  count(word) %>%
  arrange(desc(n))

# A tibble: 47,991 x 2
  word              n
  <chr>            <int>
1 anonymous        102087
2 http             29758
3 url_removed     28382
4 immigration      9034
5 covid19          6115
6 coronavirus      5190
7 china            5149
8 people           4991
9 chinesevirus     4963
10 chinavirus       4951
# ... with 47,981 more rows
```

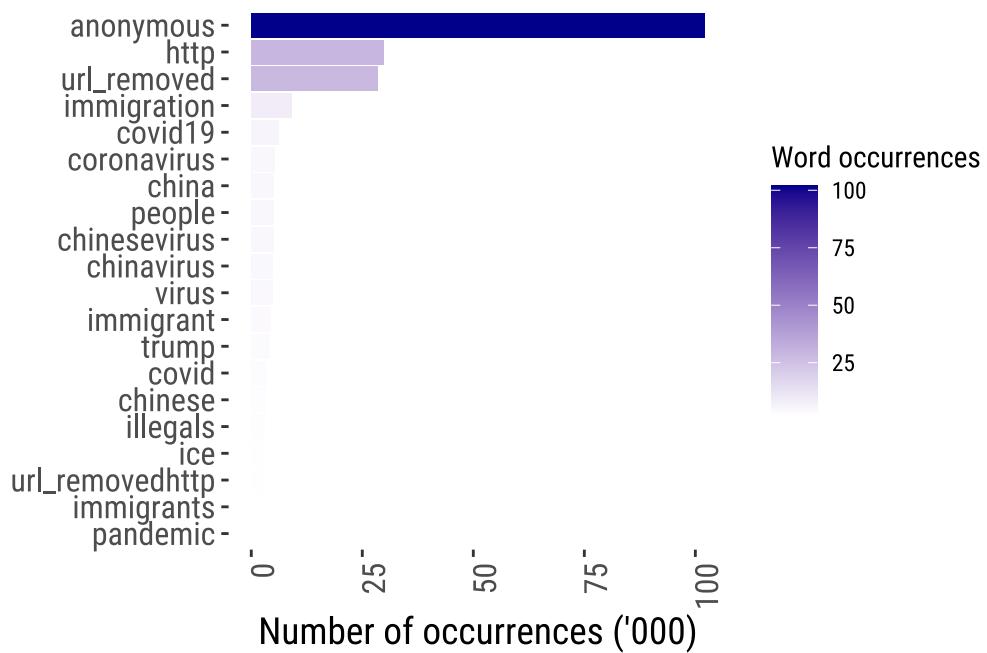
Analysing word frequencies is often the first stop in text analysis. We can easily do this using `ggplot`. Let's visualise the 20 most common words used on Twitter to express public opinions about migration-related topics.

```
tidy_tweets_df %>%
  count(word) %>%
  arrange(desc(n)) %>%
  slice(1:20) %>%
  ggplot( aes(x= reorder(word, n), y= n/1000, fill = n/1000)) +
  geom_bar( position="stack",
            stat = "identity"
            ) +
```

```

theme_tufte2() +
scale_fill_gradient(low = "white",
                     high = "darkblue") +
theme(axis.text.x = element_text(angle = 90,
                                  hjust = 1)) +
ylab("Number of occurrences ('000)") +
xlab("") +
labs(fill = "Word occurrences") +
coord_flip()

```



## 6.3 Sentiment Analysis

After pre-processing our text, we can focus on the key of this chapter; that is, measuring migration sentiment. We do this by using sentiment analysis, which as described in the introduction of this chapter, enables identifying, measuring and analysing emotional states and subjective information. It computationally infers the polarity of text, that is, whether the underpinning semantics of an opinion is positive, negative or neutral. A variety of methods and dictionaries for evaluating the opinion or emotion in text exists. We will explore four different lexicon-based approaches: `AFFIN` (REF), `bing` (REF), `nrc` (REF) and `VADER`.

### 6.3.1 Dictionary-based methods

AFFIN, bing and nrc are dictionary- or lexicon-based approaches. Sentiment lexicons include key words which are typically used to express emotions or feelings, and are assigned a numeric score for positive and negative sentiment. They may also contain scores for emotions, such as joy, anger and sadness. Sentiment lexicons can thus be used to measure the valence of a given text by searching for words that describe affect or opinion. Dictionaries can be created by examining text-based evaluations of products in online forums to ratings systems from a variety of sources. They can also be created via systematic observations about different emotions in the field of psychology or related fields.

The package `tidytext` provides access to all three lexicons. The `nrc` lexicon classifies words in a binary way into categories of positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise and trust. `nrc` constructed via Amazon Mechanical Turk (i.e. people manually labelling the emotional valence of words). The `bing` lexicon classifies words in a binary classification of positive and negative, and is based on words identified on online forums. The AFINN lexicon assigns words with a score ranging between -5 and 5 to capture the intensity of negative and positive sentiment. AFINN includes a list of sentiment-laden words used during discussions about climate change on Twitter.

 Note

Note that not all words are in the lexicons and they only contain words in the English language.

In R, we can browse the content of each lexicon using the `get_sentiment` function from `tidytext`.

 Note

Note that you may need to authorise the download of the lexicons on your console

```
get_sentiments("nrc")
```

```
# A tibble: 13,872 x 2
  word      sentiment
  <chr>     <chr>
1 abacus    trust
2 abandon   fear
3 abandon   negative
4 abandon   sadness
5 abandoned anger
```

```
6 abandoned    fear
7 abandoned negative
8 abandoned   sadness
9 abandonment anger
10 abandonment fear
# ... with 13,862 more rows
```

```
get_sentiments("bing")
```

```
# A tibble: 6,786 x 2
  word      sentiment
  <chr>     <chr>
1 2-faces    negative
2 abnormal    negative
3 abolish    negative
4 abominable negative
5 abominably negative
6 abominate   negative
7 abomination negative
8 abort      negative
9 aborted    negative
10 aborts    negative
# ... with 6,776 more rows
```

```
get_sentiments("afinn")
```

```
# A tibble: 2,477 x 2
  word      value
  <chr>     <dbl>
1 abandon    -2
2 abandoned  -2
3 abandons   -2
4 abducted   -2
5 abduction  -2
6 abductions -2
7 abhor      -3
8 abhorred   -3
9 abhorrent  -3
10 abhors    -3
# ... with 2,467 more rows
```

We can easily employ sentiment lexicons using the `get_sentiment` function from `tidytext`. Let's first create a date variable to analyse fluctuations in sentiment by day and compute sentiment scores.

```
tidy_tweets_df <- tidy_tweets_df %>%
  mutate(
    date = as.Date(substr(as.character(created_at),
                           1,
                           10))
  )

nrc_scores <- tidy_tweets_df %>%
  inner_join(get_sentiments("nrc")) %>%
  filter(sentiment %in% c("positive",
                         "negative"))
) %>%
  mutate(method = "nrc")

Joining, by = "word"

nrc_scores %>% head()

# A tibble: 6 x 5
  created_at      word      date sentiment method
  <dttm>        <chr>     <date>    <chr>    <chr>
1 2020-03-30 23:02:04 anonymous 2020-03-30 negative nrc
2 2020-03-30 23:02:04 anonymous 2020-03-30 negative nrc
3 2020-03-30 23:02:04 anonymous 2020-03-30 negative nrc
4 2020-03-30 23:02:04 anonymous 2020-03-30 negative nrc
5 2020-03-30 23:02:04 anonymous 2020-03-30 negative nrc
6 2020-03-30 23:02:04 anonymous 2020-03-30 negative nrc

bing_scores <- tidy_tweets_df %>%
  inner_join(get_sentiments("bing")) %>%
  mutate(method = "bing")

Joining, by = "word"
```

```

bing_scores %>% head()

# A tibble: 6 x 5
  created_at      word      date sentiment method
  <dttm>        <chr>    <date>   <chr>    <chr>
1 2020-03-30 23:02:04 threatening 2020-03-30 negative bing
2 2020-03-30 23:02:18 approve     2020-03-30 positive bing
3 2020-03-30 23:05:46 emergency  2020-03-30 negative bing
4 2020-03-30 23:08:48 idiots     2020-03-30 negative bing
5 2020-03-30 23:08:48 kill       2020-03-30 negative bing
6 2020-03-30 23:09:45 integral   2020-03-30 positive bing

```

```

afinn_scores <- tidy_tweets_df %>%
  inner_join(get_sentiments("afinn")) %>%
  mutate(method = "afinn")

```

Joining, by = "word"

```

afinn_scores %>% head()

# A tibble: 6 x 5
  created_at      word      date    value method
  <dttm>        <chr>    <date> <dbl> <chr>
1 2020-03-30 23:02:04 threatening 2020-03-30     -2 afinn
2 2020-03-30 23:02:04 arrested   2020-03-30     -3 afinn
3 2020-03-30 23:03:04 aboard    2020-03-30      1 afinn
4 2020-03-30 23:05:46 join     2020-03-30      1 afinn
5 2020-03-30 23:05:46 emergency 2020-03-30     -2 afinn
6 2020-03-30 23:08:48 kill     2020-03-30     -3 afinn

```

As you can see, the output of the various algorithm differs. `nrc` and `bing` provides sentiment scores classified into positive and negative. `afinn` returns a value from -5 to 5. An important feature of the three approaches explored so far is that they are based on unigrams; that is, single words. As a result, these methods do not take into account qualifiers before a word, such as in “no good” or “not true”. Additionally, these methods cannot appropriately handle negations, contractions, slang, emoticons, emojis, initialisms, acronyms, punctuation and word-shape (e.g., capitalization) as a signal of sentiment polarity and intensity (Hutto and Gilbert 2014). Most commonly, lexicon-based approaches only capture differences in sentiment polarity (i.e.,

positive or negative) but do not identify differences in sentiment intensity (strongly positive vs. moderately positive) or contrasting statements. We note that accurate identification and scoring of sarcastic statements remain a key challenge in natural language processing.

We could subtracting positive and negative score to obtain an estimate of sentiment for each day based on three lexicon approaches. The resulting data frames could be binned and used to visualise how the predominant pattern of migration sentiment changes over time. We recalculate the sentiment scores by reusing the code above and adding lines for counting, pivoting, summing and subtracting.

```
nrc_scores <- tidy_tweets_df %>%
  inner_join(get_sentiments("nrc")) %>%
    filter(sentiment %in% c("positive",
                            "negative"))
  ) %>%
  count(date, sentiment) %>%
  pivot_wider(names_from = sentiment,
              values_from = n,
              values_fill = 0) %>%
  mutate(sentiment = positive - negative) %>%
  mutate(method = "nrc") %>%
  select(date, sentiment, method)
```

Joining, by = "word"

```
bing_scores <- tidy_tweets_df %>%
  inner_join(get_sentiments("bing")) %>%
  count(date, sentiment) %>%
  pivot_wider(names_from = sentiment,
              values_from = n,
              values_fill = 0) %>%
  mutate(sentiment = positive - negative) %>%
  mutate(method = "bing") %>%
  select(date, sentiment, method)
```

Joining, by = "word"

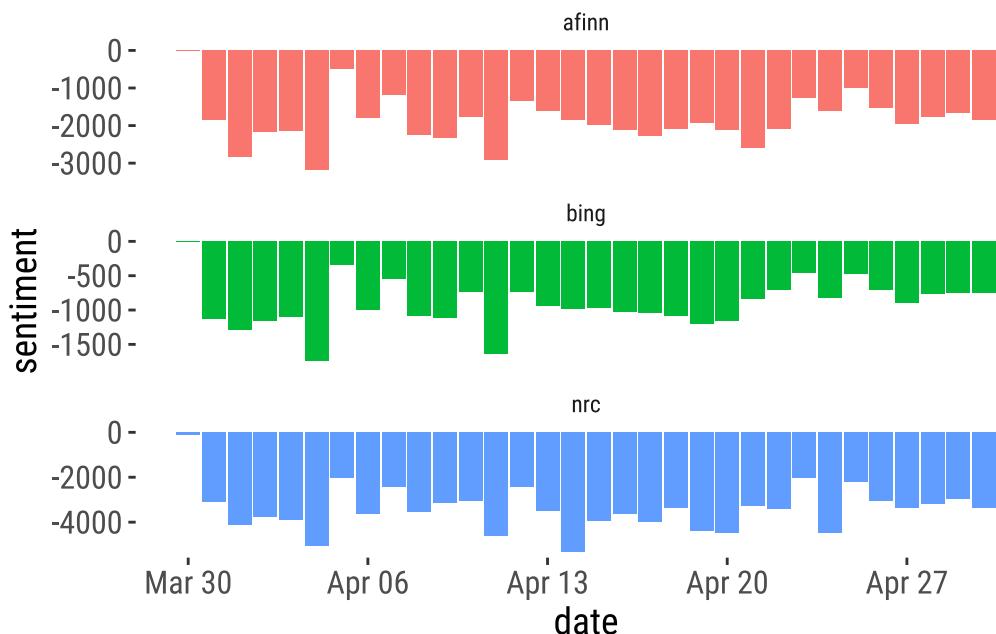
```
afinn_scores <- tidy_tweets_df %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(date) %>%
```

```
summarise(sentiment = sum(value)) %>%
mutate(method = "afinn")
```

Joining, by = "word"

Once we have daily sentiment scores, we bin them together and visualise them.

```
bind_rows(nrc_scores,
          bing_scores,
          afinn_scores) %>%
ggplot( aes(x = date, y = sentiment, fill = method)) +
geom_col(show.legend = FALSE) +
theme_tufte2() +
facet_wrap(~method, ncol = 1, scales = "free_y")
```



All three lexicons display the same predominant pattern of increasing negative migration sentiment between March and April. They differ in the representation they provide during earlier months. While **afinn** and **bing** lexicons concur in suggesting that a negative sentiment was the prevalent pattern of sentiment towards migration during these months, **nrc** paints a different picture of a predominantly positive sentiment.

### 6.3.2 VADER

We move on to explore VADER. VADER is a lexicon and rule-based sentiment analysis tool which is tailored to the analysis of sentiments expressed in social media, and stands for Valence Aware Dictionary and sEntiment Reasoner (Hutto and Gilbert 2014). VADER has been shown to perform better than 11 typical state-of-practice sentiment algorithms at identifying the polarity expressed in tweets (Hutto and Gilbert 2014), and has remained one of the most widely used sentiment analysis methods for social media data (e.g. Elbagir and Yang 2020) ). See Ghani et al. (2019) and Rosa et al. (2019) for recent comprehensive reviews of social media analytics.

VADER overcomes limitations of existing approaches (Hutto and Gilbert 2014). It also captures differences in sentiment intensity, contrasting statements, and can handle complex sentences, including typical negations (e.g. “not good”), contractions (e.g. “wasn’t very good”), conventional use of punctuation to signal increased sentiment intensity (e.g. “Good!!!”), use of word-shape to signal emphasis (e.g. using ALL CAPS), using degree modifiers to alter sentiment intensity (e.g. intensity boosters (e.g. “very”) and intensity dampeners (e.g.”kind of”), sentiment-laden slang (e.g. ‘sux’), slang words as modifiers (e.g. ‘uber’ or ‘friggin’ or ‘kinda’), emoticons (:) and :D), translating utf-8 encoded emojis ( , and ), initialisms and acronyms (e.g. ‘lol’). VADER can also handle entire sentences or ngrams, rather than only unigrams. See some examples below.

```
vader_df("wasn't very good")
```

```
text      word_scores compound pos    neu    neg but_count
1 wasn't very good {0, 0, -1.62282}   -0.386    0 0.433 0.567       0
```

```
vader_df("not good")
```

```
text word_scores compound pos    neu    neg but_count
1 not good {0, -1.406}   -0.341    0 0.294 0.706       0
```

```
vader_df("good")
```

```
text word_scores compound pos neu neg but_count
1 good      {1.9}     0.44   1   0   0       0
```

```
vader_df("Good!!!!")
```

```

text word_scores compound pos neu neg but_count
1 Good!!!      {1.9}    0.583   1   0   0       0

vader_df("VERY good!!!")

text word_scores compound pos neu neg but_count
1 VERY good!!! {0, 2.926}  0.701 0.828 0.172   0

vader_df("wasn't bad but very good")

text          word_scores compound pos neu neg
1 wasn't bad but very good {0, 0.925, 0, 0, 3.2895}  0.736 0.674 0.326 0
but_count
1           1

```

The output is a vector with the following entries:

- \* *word\_scores*: a string that contains an ordered list with the matched scores for each of the words in the text. For the first example, you can see three scores i.e. a 0 score for “wasn’t” and “very” and a negative score for “good” reflecting the meaning of “good” in the text.
- \* *compound*: the resulting valence compound of VADER for the entire text after applying modifiers and aggregation rules.
- \* *pos, neg, and neu*: the parts of the compound for positive, negative, and neutral content. These take into account modifiers and are combined when calculating the compound score
- \* *but\_count*: an additional count of “but” since it can complicate the calculation of sentiment.

Let’s think about the results from the examples above, what is the piece of text with the most negative and positive sentiment score? Why? How do modifiers, amplifiers and negators influence the meaning of text?

Now we will use VADER to explore the sentiment towards migration during the wake of the COVID-19 pandemic. To reduce computational requirements, we will work with a sub-sample of our data to obtain sentiment scores at the tweet level. This is unlike our previous analysis which returned word-level scores. Obtaining sentiment scores via VADER may take some time. So do not panic, relax and wait. If this is taking too long, you can use the ‘`tweet_vader_scores.rds`’ in the data folder for this chapter.

Note that for this example we will use the tweet text as VADER can handle various of the issues that would be a problem using the previous three approaches (as we have described above). Nonetheless, for your our work you may want to explore the influence of regular expressions.

```

# tweet level
tweet_vader_scores <- tweets_df %>%
  select(text) %>%
  vader_df(text)

# combine vader scores,tweet ids and dates
tweets_scores_df <- cbind(tweets_df$id, tweets_df$created_at, tweet_vader_scores)

# rename vars and extract day var
tweets_scores_df <- tweets_scores_df %>%
  rename(
    id = "tweets_df$id",
    created_at = "tweets_df$created_at"
  ) %>%
  dplyr::mutate(
    date = as.Date(substr(as.character(created_at),
      1,
      10))
  )

```

### *Concentration*

We have done the hard work of computing sentiment scores. We can now start analysing the results. As any exploratory analysis, a first feature you may want to analyse if the overall distribution of sentiment scores. Applied to public opinion data, such analysis may give you an idea of how socially polarised is a discussion on social media. To this end, we can create a histogram.

```

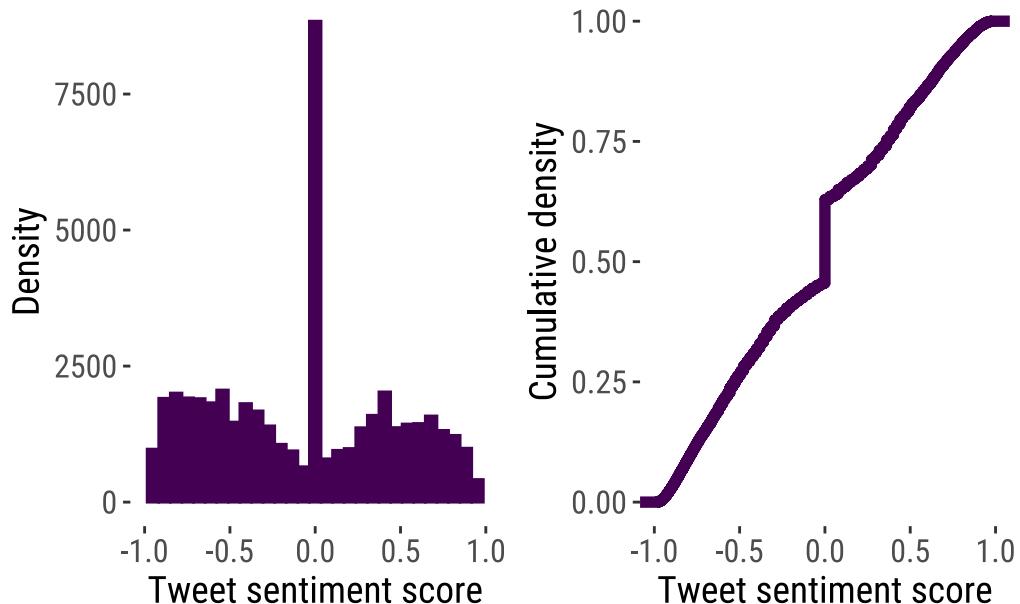
p1 <- ggplot(data = tweets_scores_df) +
  geom_histogram(aes(x = compound,
                     binwidth = 0.05),
                 fill = "#440154FF",
                 color="#440154FF") +
  theme_tufte2() +
  labs(x= "Tweet sentiment score",
       y = "Density")

p2 <- ggplot(tweets_scores_df, aes(compound)) +
  stat_ecdf(geom = "step",
            size = 2,
            colour = "#440154FF") +
  theme_tufte2() +

```

```
  labs(x= "Tweet sentiment score",
       y = "Cumulative density")
```

```
p1 | p2
```



We produce two plots exploring the frequency and cumulative distribution of migration sentiment scores. The results indicate that a concentration around zero and also at both extremes i.e. below -0.5 and over 0.5, suggesting that migration is very polarising social issue.

#### *Temporal evolution*

We can also explore the temporal evolution of sentiment towards migration over time. The results indicate that migration sentiment remained slightly negative but stable during March to April 2020.

```
# plot sentiment scores by day
p3 <- ggplot(tweets_scores_df,
              aes(x = date, y = compound)) +
  geom_point(colour = "gray", alpha = 0.3, size = 1, shape=".") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red", size = .3) +
  geom_smooth(method = "loess", se = FALSE, size=2, span = 0.3, color="#440154FF") +
  theme_tufte2() +
```

```

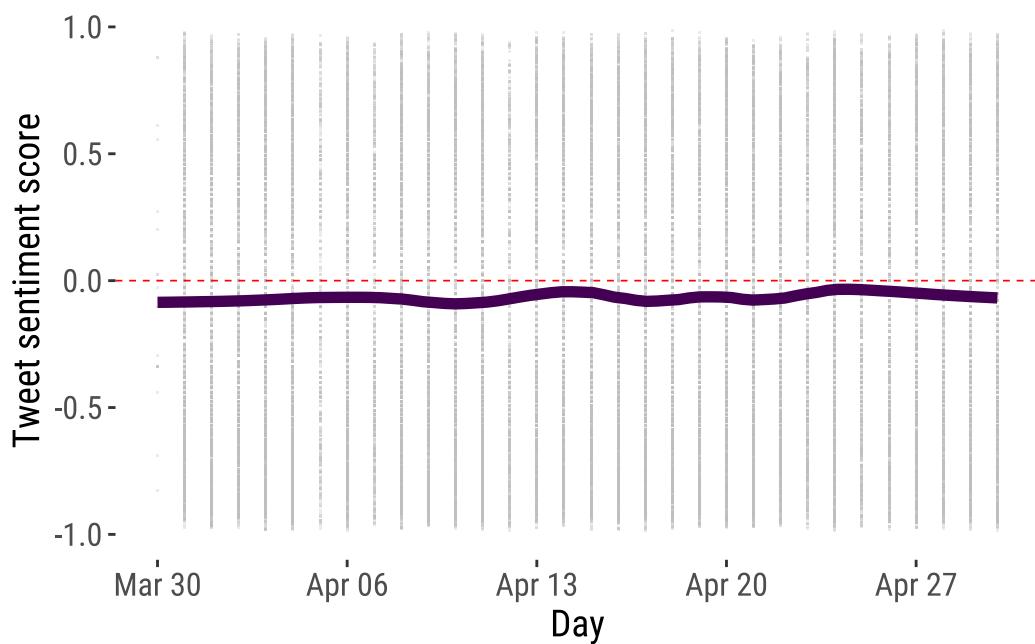
  labs(x= "Day",
       y = "Tweet sentiment score") +
  scale_y_continuous(limits = c(-1, 1))

```

p3

Warning: Removed 1 rows containing non-finite values (`stat\_smooth()`).

Warning: Removed 1 rows containing missing values (`geom\_point()`).



### *Composition*

Analysing the compound conceals the composition of sentiment in tweets, particularly potential rises in strongly negative sentiment. We then analyse the composition of tweets classifying our sentiment score into “Strongly Negative”, “Negative”, “Neutral”, “Positive” and “Strongly Positive” as shown below. The results indicate that the composition remained largely stable over time with 50% of all tweets being negative, of which around 25% were strongly negative. In contrast, less than 20% of all tweets were strongly positive. These results suggest that anti-migration sentiment tends to use a stronger rhetoric than pro-migration sentiment.

```

# sentiment categories
tweets_scores_df <- tweets_scores_df %>%
  mutate(stance_group =
    case_when(
      compound >= -.05 & compound <= .05 ~ 3,
      compound < -.5 ~ 1,
      compound < -.05 & compound >= -.5 ~ 2,
      compound > .05 & compound <= .5 ~ 4,
      compound > .5 ~ 5,
    )
  )

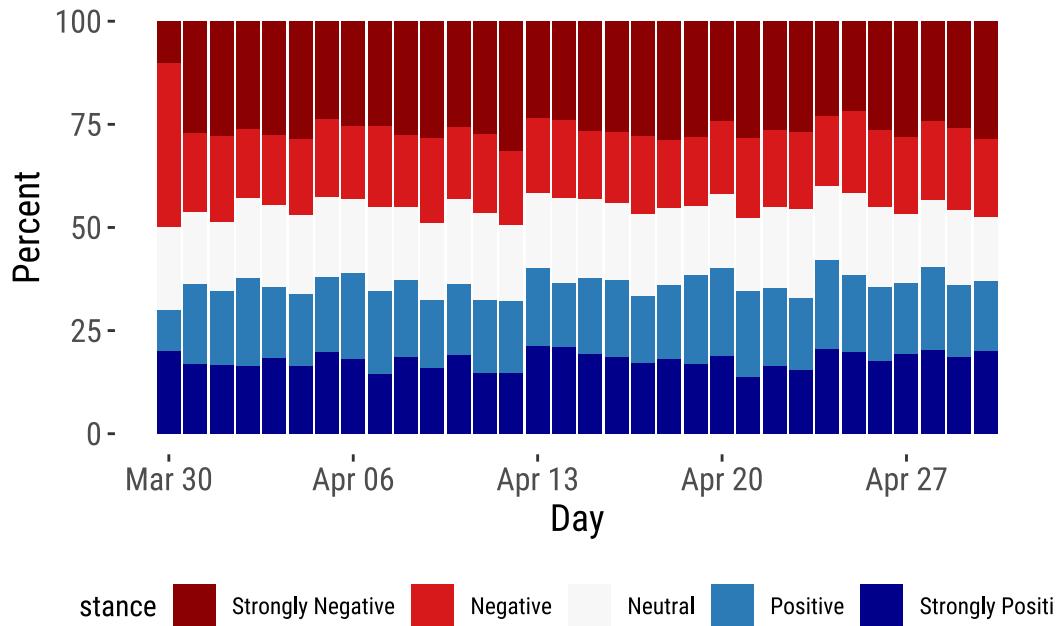
# count in each sentiment category by day
composition_tab <- tweets_scores_df %>%
  group_by(date) %>%
  dplyr::count(date, stance_group) %>%
  spread(stance_group, n)

# percentage in each sentiment category by day
composition_percent_tab <- cbind(composition_tab[,1], (composition_tab[,2:6] / rowSums(composition_tab[,2:6])) [,])
  .[, c(1, 3, 4, 2, 5, 6)] %>%
  gather(stance, percent, -date)

# composition of sentiment score by day
p4 <- ggplot(composition_percent_tab ,
  aes(fill = stance, y = percent, x = date)) +
  geom_bar(position="stack", stat="identity") +
  theme_tufte2() +
  theme(legend.position = "bottom") +
  scale_fill_manual(values = c("darkred", "#d7191c", "#f7f7f7", "#2c7bb6", "darkblue"),
                    labels = c("Strongly Negative", "Negative", "Neutral", "Positive", "Very Positive"))
  labs(x= "Day",
       y = "Percent")

```

p4



#### *i* Note

Natural language processing is a rapidly evolving field and various new sentiment analysis algorithms emerge over the last five years. They are often tailored to address specific tasks and their performance can vary widely across datasets. So before deciding on a particular algorithm, consult the literature on what has been previously used to identify standard approaches their limitations and strengths.

## 6.4 Questions

For the second assignment, we will focus on the United Kingdom as our geographical area of analysis. We will use a dataset of tweets about migration posted by users in the United Kingdom during February 24th 2021 to July 1st 2022. This period coincides with the start of the war in Ukraine and is expected to capture changes in migration sentiment. The dataset contains the following information:

- tweet\_id: unique tweet identifier
- created\_at: date tweet were posted
- place\_name: name of place linked to tweet
- lat: latitude

- long: longitude
- text: text content of tweet

```
tweets_qdf <- readRDS("./data/sentiment-analysis/uk_tweets_24022021_01072022.rds")
glimpse(tweets_qdf)
```

```
Rows: 34,490
Columns: 6
$ tweet_id    <dbl> 1.364707e+18, 1.364694e+18, 1.364692e+18, 1.364684e+18, 1.3~ 
$ created_at   <dttm> 2021-02-24 22:40:21, 2021-02-24 21:47:44, 2021-02-24 21:42~ 
$ place_name   <chr> "Westhumble", "Rushden", "Birmingham", "Cardiff", "Alexandr~ 
$ lat          <dbl> -0.3302450, -0.6038262, -1.8906405, -3.1797998, -4.5748715, ~ 
$ long         <dbl> 51.25447, 52.28951, 52.49397, 51.49700, 55.98702, 53.64739, ~ 
$ text         <chr> "@post Liberal Voting for Griffin was part of this working ~
```

Using VADER:

1. Obtain sentiment scores and create plots to visualise the overall distribution of sentiment scores;
2. Create plots to analyse the temporal evolution of sentiment over time.
3. Visualise the geographical patterns of sentiment scores - see the spatial autocorrelation section on Rowe (2022a) to map sentiment scores using `ggplot`.

Analyse and discuss: a) the extent of anti-immigration sentiment in the United Kingdom; b) how it has changes over time in intensity and composition; and, c) the degree of spatial concentration in anti-immigration sentiment in the country.

## **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

- Abbott, Andrew, and Angela Tsay. 2000. "Sequence Analysis and Optimal Matching Methods in Sociology: Review and Prospect." *Sociological Methods & Research* 29 (1): 3–33.
- 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>.
- Backman, Mikaela, Esteban Lopez, and Francisco Rowe. 2020. "The Occupational Trajectories and Outcomes of Forced Migrants in Sweden. Entrepreneurship, Employment or Persistent Inactivity?" *Small Business Economics* 56 (3): 963–83. <https://doi.org/10.1007/s11187-019-00312-z>.
- Bail, Christopher A., Lisa P. Argyle, Taylor W. Brown, John P. Bumpus, Haohan Chen, M. B. Fallin Hunzaker, Jaemin Lee, Marcus Mann, Friedolin Merhout, and Alexander Volfovsky. 2018. "Exposure to Opposing Views on Social Media Can Increase Political Polarization." *Proceedings of the National Academy of Sciences* 115 (37): 9216–21. <https://doi.org/10.1073/pnas.1804840115>.
- Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer.
- Cabrera-Arnau, Carmen, Chen Zhong, Michael Batty, Ricardo Silva, and Soong Moon Kang. 2022. "Inferring Urban Polycentricity from the Variability in Human Mobility Patterns." arXiv. <https://doi.org/10.48550/ARXIV.2212.03973>.
- 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>.
- Cheong, Pauline Hope, Rosalind Edwards, Harry Goulbourne, and John Solomos. 2007. "Immigration, Social Cohesion and Social Capital: A Critical Review." *Critical Social Policy* 27 (1): 24–49. <https://doi.org/10.1177/0261018307072206>.
- Coates, Melanie. 2020. "Covid-19 and the Rise of Racism." *BMJ*, April, m1384. <https://doi.org/10.1136/bmj.m1384>.
- Cowper, Andy. 2020. "Covid-19: Are We Getting the Communications Right?" *BMJ*, March, m919. <https://doi.org/10.1136/bmj.m919>.
- 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>.

- Elbagir, Shihab, and Jing Yang. 2020. "Sentiment Analysis on Twitter with Python's Natural Language Toolkit and VADER Sentiment Analyzer." *IAENG Transactions on Engineering Sciences*, January. [https://doi.org/10.1142/9789811215094\\_0005](https://doi.org/10.1142/9789811215094_0005).
- European Commision. 2019. "10 Trends Shaping Migration." <https://op.europa.eu/en/publication-detail/-/publication/aa25fb8f-10cc-11ea-8c1f-01aa75ed71a1>.
- Fielding, A. J. 1992. "Migration and Social Mobility: South East England as an Escalator Region." *Regional Studies* 26 (1): 1–15. <https://doi.org/10.1080/00343409212331346741>.
- Franklin, Rachel. 2022. "Quantitative Methods II: Big Theory." *Progress in Human Geography* 47 (1): 178–86. <https://doi.org/10.1177/03091325221137334>.
- Gabadinho, Alexis, Gilbert Ritschard, Nicolas S. Müller, and Matthias Studer. 2011. "Analyzing and Visualizing State Sequences in R with TraMineR." *Journal of Statistical Software* 40 (4). <https://doi.org/10.18637/jss.v040.i04>.
- Gabadinho, Alexis, Gilbert Ritschard, Matthias Studer, and Nicolas S Müller. 2009. "Mining Sequence Data in r with the TraMineR Package: A User's Guide." *Geneva: Department of Econometrics and Laboratory of Demography, University of Geneva*.
- Ghani, Norjihan Abdul, Suraya Hamid, Ibrahim Abaker Targio Hashem, and Ejaz Ahmed. 2019. "Social Media Big Data Analytics: A Survey." *Computers in Human Behavior* 101 (December): 417–28. <https://doi.org/10.1016/j.chb.2018.08.039>.
- González-Leonardo, Miguel, Niall Newsham, and Francisco Rowe. 2023. "Understanding Population Decline Trajectories in Spain Using Sequence Analysis." *Geographical Analysis*, January. <https://doi.org/10.1111/gean.12357>.
- 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](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>.
- Home Affairs Committee. 2020. "Oral evidence: Home Office preparedness for Covid-19 (Coronavirus), HC 232." London: House of Commons. <https://committees.parliament.uk/oralevidence/359/default/>.
- Hutto, C., and Eric Gilbert. 2014. "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text." *Proceedings of the International AAAI Conference on Web and Social Media* 8 (1): 216–25. <https://doi.org/10.1609/icwsm.v8i1.14550>.
- 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>.
- Kaufman, Leonard, and Peter J Rousseeuw. 2009. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons.
- 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>.
- Newman, Mark. 2018. *Networks / Mark Newman*. Second edition. Oxford: Oxford University Press.
- Newsham, Niall, and Francisco Rowe. 2022a. “Understanding the Trajectories of Population Decline Across Rural and Urban Europe: A Sequence Analysis.” <https://doi.org/10.48550/ARXIV.2203.09798>.
- . 2022b. “Understanding Trajectories of Population Decline Across Rural and Urban Europe: A Sequence Analysis.” *Population, Space and Place*, December. <https://doi.org/10.1002/psp.2630>.
- Ognyanova, K. 2016. “Network Analysis with r and Igraph: NetSci x Tutorial.” [www.kateto.net/networks-r-igraph](http://www.kateto.net/networks-r-igraph).
- Patias, Nikos, Francisco Rowe, and Dani Arribas-Bel. 2021. “Trajectories of Neighbourhood Inequality in Britain: Unpacking Inter-Regional Socioeconomic Imbalances, 1971-2011.” *The Geographical Journal* 188 (2): 150–65. <https://doi.org/10.1111/geoj.12420>.
- Patias, Nikos, Francisco Rowe, Stefano Cavazzi, and Dani Arribas-Bel. 2021. “Sustainable Urban Development Indicators in Great Britain from 2001 to 2016.” *Landscape and Urban Planning* 214 (October): 104148. <https://doi.org/10.1016/j.landurbplan.2021.104148>.
- 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>.
- Prieto Curiel, Rafael, Carmen Cabrera-Arnau, and Steven Richard Bishop. 2022. “Scaling Beyond Cities.” *Frontiers in Physics* 10. <https://doi.org/10.3389/fphy.2022.858307>.
- 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>.
- Rosa, H., N. Pereira, R. Ribeiro, P. C. Ferreira, J. P. Carvalho, S. Oliveira, L. Coheur, P. Paulino, A. M. Veiga Simão, and I. Trancoso. 2019. “Automatic Cyberbullying Detection: A Systematic Review.” *Computers in Human Behavior* 93 (April): 333–45. <https://doi.org/10.1016/j.chb.2018.12.021>.
- Rowe, Francisco. 2021a. “Using Twitter Data to Monitor Immigration Sentiment.” <http://dx.doi.org/10.31219/osf.io/sf7u4>.
- . 2021b. “Big Data and Human Geography.” <http://dx.doi.org/10.31235/osf.io/phz3e>.
- . 2022a. “Introduction to Geographic Data Science.” *Open Science Framework*, August. <https://doi.org/10.17605/OSF.IO/VHY2P>.

- . 2022b. “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, and Dani Arribas-Bel. 2022. “Spatial Modelling for Data Scientists.” *Open Science Framework*, August. <https://doi.org/10.17605/OSF.IO/8F6XR>.
- Rowe, Francisco, Jonathan Corcoran, and Martin Bell. 2016. “The Returns to Migration and Human Capital Accumulation Pathways: Non-Metropolitan Youth in the School-to-Work Transition.” *The Annals of Regional Science* 59 (3): 819–45. <https://doi.org/10.1007/s00168-016-0771-8>.
- Rowe, Francisco, Michael Mahony, Eduardo Graells-Garrido, Marzia Rango, and Niklas Sievers. 2021. “Using Twitter to Track Immigration Sentiment During Early Stages of the COVID-19 Pandemic.” *Data & Policy* 3. <https://doi.org/10.1017/dap.2021.38>.
- Rowe, Francisco, Michael Mahony, Niklas Sievers, Marzia Rango, and Eduardo Graells-Garrido. 2021. “Sentiment towards Migration during COVID-19. What Twitter Data Can Tell Us.” *IOM Publications*.
- 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>.
- Salmela-Aro, Katriina, Noona Kiuru, Jari-Erik Nurmi, and Mervi Eerola. 2011. “Mapping Pathways to Adulthood Among Finnish University Students: Sequences, Patterns, Variations in Family- and Work-Related Roles.” *Advances in Life Course Research* 16 (1): 25–41. <https://doi.org/10.1016/j.alcr.2011.01.003>.
- 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>.
- “Stop the Coronavirus Stigma Now.” 2020. *Nature* 580 (7802): 165–65. <https://doi.org/10.1038/d41586-020-01009-0>.
- Tatem, Andrew J. 2017. “WorldPop, Open Data for Spatial Demography.” *Scientific Data* 4 (1). <https://doi.org/10.1038/sdata.2017.4>.
- Turok, Ivan, and Vlad Mykhnenko. 2007. “The Trajectories of European Cities, 1960–2005.” *Cities* 24 (3): 165–82. <https://doi.org/10.1016/j.cities.2007.01.007>.
- 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>.