Population Science

Francisco Rowe, Carmen Cabrera-Arnau, Elisabetta Pietrostefani

2024-01-30

Table of contents

Welcome **6**
Contact . . . 6

1 Overview **7**
1.1 Aims . . . 7
1.2 Learning Outcomes . . . 7
1.3 Feedback . . . 8
1.4 Computational Environment . . . 8
1.4.1 List of libraries . . . 9
1.5 Assessment . . . 10
1.5.1 Format Requirements . . . 11

2 Introducing Population Science **13**
2.1 Introduction . . . 13
2.2 Defining digital footprint data . . . 14
2.3 Opportunities of digital footprint data . . . 15
2.4 Challenges of digital footprint data . . . 17
2.4.1 Conceptual challenges . . . 17
2.4.2 Methodological challenges . . . 17
2.4.3 Ethical challenges . . . 18
2.5 Conclusion . . . 19

3 Geodemographics **20**
3.1 Dependencies . . . 20
3.2 Data . . . 21
3.2.1 Land use data for Greater Manchester . . . 21
3.2.2 Import the data . . . 22
3.3 Preparing the data for GDC . . . 22
3.3.1 Choice of geographic units . . . 22
3.3.2 Variables of interest . . . 23
3.4 Standardisation . . . 27
3.4.1 Across geographic units . . . 27
3.4.2 Variable standardisation . . . 28
3.5 Checking for variable association . . . 28

3.6 The clustering process . . . 30
3.6.1 K-means . . . 30
3.6.2 Number of clusters . . . 31
3.6.3 Other clustering methods . . . 33
3.7 GDC results . . . 33
3.7.1 Mapping the clusters . . . 33
3.7.2 Cluster interpretation . . . 35
3.7.3 Testing . . . 36
3.8 Questions . . . 37

4 Sequence Analysis **39**
4.1 Dependencies . . . 39
4.2 Data . . . 40
4.2.1 Data wrangling . . . 46
4.2.2 Exploratory data analysis . . . 48
4.3 Application . . . 53
4.3.1 Defining outcome process . . . 54
4.3.2 Optimal matching . . . 55
4.3.3 Clustering . . . 63
4.3.4 Visualising . . . 64
4.4 Questions . . . 69

5 Network Analysis **71**
5.1 Dependencies . . . 71
5.2 Data . . . 72
5.2.1 The US Census dataset . . . 72
5.2.2 Import the data . . . 72
5.3 Creating networks . . . 73
5.3.1 Starting from the basics . . . 73
5.3.2 Adding attributes . . . 76
5.4 Reading networks from data files . . . 78
5.4.1 Preparing the data to create an igraph object . . . 78
5.4.2 Filtering the data to create a subgraph . . . 80
5.5 Network visualisation . . . 82
5.5.1 Visualisation with igraph . . . 82
5.5.2 Visualisation of spatial networks . . . 84
5.5.3 Alternative visualisations . . . 87
5.6 Network metrics . . . 87
5.6.1 Density . . . 87
5.6.2 Reciprocity . . . 88
5.6.3 Degree . . . 88
5.6.4 Distances . . . 90
5.6.5 Centrality . . . 91

5.6.6 Hubs and authorities . . . 94
5.7 Questions . . . 96

6 Sentiment Analysis **99**
6.1 Dependencies . . . 100
6.2 Data . . . 100
6.2.1 Text data structures . . . 101
6.2.2 Basic text data principles . . . 102
6.3 Sentiment Analysis . . . 111
6.3.1 Dictionary-based methods . . . 112
6.3.2 VADER . . . 118
6.4 Questions . . . 125

7 Topic Modelling **127**
7.1 Dependencies . . . 128
7.2 Data . . . 129
7.2.1 Text data structures . . . 131
7.2.2 Basic text data principles . . . 132
7.3 Topic Modelling . . . 135
7.3.1 Word-topic probabilities . . . 135
7.3.2 Greatest differences in β . . . 137
7.3.3 Structural Topic Modelling . . . 139
7.3.4 Limitations of Topic Models . . . 142
7.4 Questions . . . 143

8 Modelling Time **145**
8.1 Dependencies . . . 145
8.2 Data . . . 146
8.3 Modelling Time . . . 146
8.4 Modelling Time and Space . . . 158
8.5 Questions . . . 163

9 Assessing Interventions **165**
9.1 Dependencies . . . 166
9.2 Data . . . 166
9.3 Data Exploration . . . 168
9.4 Difference in Difference . . . 173
9.5 Questions . . . 181

10 Machine Learning **183**
10.1 Dependencies . . . 184
10.2 Data . . . 184
10.3 Splitting the data . . . 185

10.4 Decision trees . . . 188
10.4.1 Fitting the training data . . . 188
10.4.2 Measuring the performance of regression models . . . 192
10.4.3 Bagging . . . 193
10.5 Random forests . . . 195
10.5.1 Basic implementation . . . 196
10.5.2 Tuning . . . 199
10.6 Questions . . . 201

11 Data sets **203**
11.1 Greater Machester land use data . . . 203
Availability . . . 203
Variables . . . 203
Source & Pre-processing . . . 203
11.2 British administrative boundaries (LSOAs and LAs) . . . 204
Availability . . . 204
Variables . . . 204
Projection . . . 206
Source & Pre-processing . . . 208
11.3 Worldpop population count data for Ukraine . . . 208
11.4 Census population count data for UK . . . 208
11.5 Ukraine's administrative boundaries . . . 208
11.6 Internal migration flows between US metropolitan areas and between London boroughs . . . 208
Availability . . . 208
Variables . . . 209
Source & pre-processing . . . 210
11.7 Twitter data on public opinion originated in the US and in the UK . . . 210
11.8 Reddit data . . . 210
11.9 Google mobility data for Italy and the UK . . . 210
11.10 COVID-19 cases data for London and Rome . . . 210
11.11 Census MSOA data for England and Wales . . . 210
Availability . . . 210
Variables . . . 211
Source & pre-processing . . . 212

References **213**

Welcome

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

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

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

Contact

- **Module lead:** ***Dr Carmen Cabrera-Arnau*** - c.cabrera-arnau [at] liverpool.ac.uk - Lecturer in Geographic Data Science
- *Prof Francisco Rowe* - f.rowe-gonzalez [at] liverpool.ac.uk - Professor in Population Data Science
- *Dr Elisabetta Pietrostefani* - e.pietrostefani [at] liverpool.ac.uk - Lecturer in Geographic Data Science

  Find us in Roxby Building, University of Liverpool, UK

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-4.3.2
- RStudio 2023.12.0-369
- Quarto 1.4.543
- 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"
- "rpart"
- "rpart.plot"
- "ggplot2"
- "Metrics"
- "caret"
- "randomForest"
- "ranger"
- "wpgpDownloadR"
- "devtools"
- "ggseqplot"
- "tidytext"
- "tm"
- "textdata"
- "topicmodels"
- "RedditExtractoR"
- "stm"
- "dygraphs"

- "plotly"
- "ggpmisc"
- "ggformula"
- "ggimage"
- "modelsummary"
- "gtools"
- "webshot"
- "gridExtra"
- "broom"
- "rtweet"

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

```
# package names
packages <- c( "tidyverse", "viridis", "viridisLite", "ggthemes", "patchwork", "showtext",

# install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

# packages loading
invisible(lapply(packages, library, character.only = TRUE))
```

1.5 Assessment

The final module mark is composed of the *two computational essays*. Together they are designed to cover the materials introduced in the entirety of content covered during the semester. A computational essay is an essay whose narrative is supported by code and computational results that are included in the essay itself. Each teaching week, you will be required to address a set of questions relating to the module content covered in that week, and to use the material that you will produce for this purpose to build your computational essay.

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

Assignment 2 (50%) refers to the set of questions at the end of Chapter 6, Chapter 7, Chapter 8, Chapter 9 and Chapter 10. You are required to use your responses to build your computational

essay. Each chapter provides more specific guidance of the tasks and discussion that you are required to consider in your assignment.

1.5.1 Format Requirements

Both assignments will have the same requirements:

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

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

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

Submission is electronic only via Turnitin on Canvas.

1.5.1.1 Marking criteria

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

- 0-15: no documentation and use of default options.
- 16-39: little documentation and use of default options.
- 40-49: some documentation, and use of default options.
- 50-59: extensive documentation, and edit of some of the options provided in the notebook (e.g. change north arrow location).
- 60-69: extensive well organised and easy to read documentation, and evidence of understanding of options provided in the code (e.g. tweaking existing options).
- 70-79: all above, plus clear evidence of code design skills (e.g. customising graphics, combining plots (or tables) into a single output, adding clear axis labels and variable names on graphic outputs, etc.).

- 80-100: all as above, plus code containing novel contributions that extend/improve the functionality the code was provided with (e.g. comparative model assessments, novel methods to perform the task, etc.).

2 Introducing Population Science

2.1 Introduction

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

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

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

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

Yet, the use of digital footprint data also poses major conceptual, methodological and ethical challenges (Rowe 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).

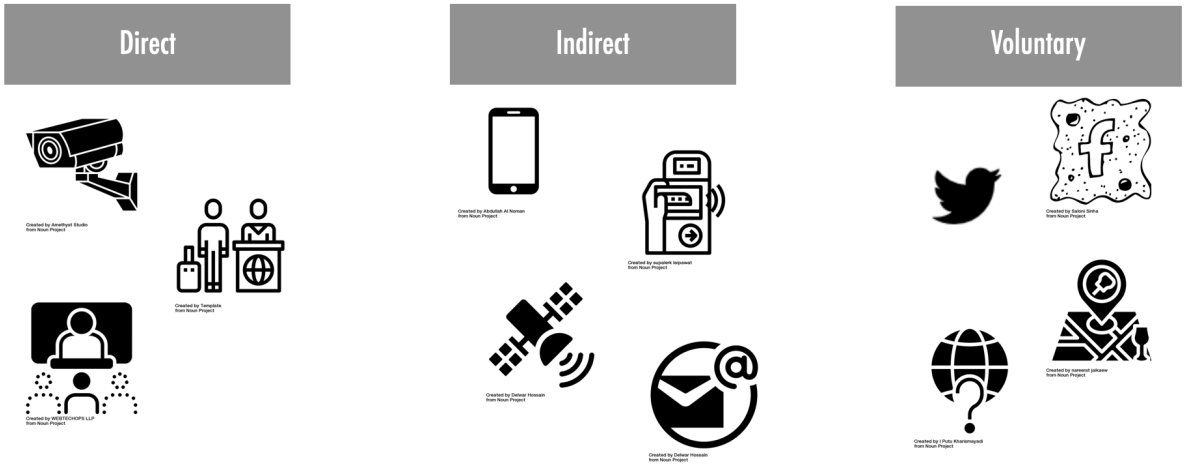


Figure 2.1: Digital footprint systems

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 steams of time-stamped location data. Such information thus provides an opportunity to trace and enhance our understanding human populations over highly granular spatial scales and time intervals, going beyond the static representation afforded by most traditional data sources. Spatial human interactions, and how people use and are influenced by their environment, can be analysed in a temporally dynamic way.

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

Digital footprint data are generated in *real-time*. Unlike traditional systems of data collection and release, digital footprint data can be streamed continuously in real- or near real-time. Commercial transactions are generally recorded on bank ledgers as bank card payments occur at retail shops. Individual mobile phone's location are captured as applications ping cellular antennas. Such information offer an opportunity to monitor and response to rapidly evolving situations, such as the COVID-19 pandemic (Green, Pollock, and Rowe 2021), natural disasters (Rowe 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 oppotunites to enhance our understanding of population processes and support individual, business and government decisions to improve targeted processes and outcomes. Businesses have used digital footprint data to segment their consumer populations and improve their targeting of marketing content, products and ultimately increase sales and revenue (Dolega, Rowe, and Branagan 2021). Governments and health care institutions, particularly during the COVID-19 have leverage digital footprint data to monitor the spread of the pandemic and develop appropriate mitigation responses (Green, Pollock, and Rowe 2021). However, the use of digital footprint data poses major conceptual, methodological and ethical challenges - which need to be overcome to unleash their full potential. The aim of this book is to address of the key methodological challenges. In particular, this book seeks to provide applied training on the practical application of commonly used machine learning and artificial intelligence approaches to leverage on digital footprint data in the understanding of human behaviour and population processes.

3 Geodemographics

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

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

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

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

3.1 Dependencies

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

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

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

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

```
[1] "/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202324/r4ps"
```

3.2 Data

3.2.1 Land use data for Greater Manchester

Land use patterns offer insights into the demographic characteristics and needs of a population within a specific area. How residents use their land reflects their preferences, demands, and activities. The extent of residential land indicates the size, density, and distribution of the population, as well as housing preferences and household sizes. The extent of commercial and industrial land use relates to economic activities, employment opportunities, and the overall economic vitality of a region. Agricultural land use might indicate the food demands and production capabilities of the population. Additionally, land designated for public services, education, healthcare, and recreation showcases the amenities and social infrastructure necessary to support the population's needs and quality of life. Classifying regions by their similarity in their land use can therefore provide valuable insights into the lifestyle, economic status, and social requirements of the residents in a given area.

In this Chapter we will be looking at land use data provided by What do 'left behind' areas look like over time?, a project created by the Geographic Data Science Lab at the University of Liverpool to host open reproducible code and resulting data for the Briefing: "What do 'left behind' areas look like over time?". The land use data is based on digital data from the CORINE Land Cover (CLC) dataset. This well-known dataset is compiled and maintained by the European Environment Agency (EEA) and derived from satellite imagery and aerial photographs, using remote sensing technology and image analysis techniques.

In particular, we use the file `manchester_land_cover_2011.gpkg` based on the file `lsoa_land_cover.csv` available here, which contains land use data for each Lower Layer Super Output Area (LSOA) in Greater Manchester.

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, `manchester_land_cover_2011.gpkg` as a simple feature object and call it `sf_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 is included in the `sf_LSOA` variable, in the column called `geom`.

```
# The raw data can be obtained from link below, but it has been cleaned by Carmen Cabrera-
# https://github.com/GDSL-UL/APPG-LBA/blob/main/data/lsoa_land_cover.csv

sf_LSOA <- st_read("./data/geodemographics/manchester_land_cover_2011.gpkg")
```

```
Reading layer `manchester_land_cover_2011' from data source
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202324/r4ps/data/geodemographi
  using driver `GPKG'
Simple feature collection with 1673 features and 44 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 351662.3 ymin: 381166 xmax: 406087.2 ymax: 421037.7
Projected CRS: OSGB36 / British National Grid
```

3.3 Preparing the data for GDC

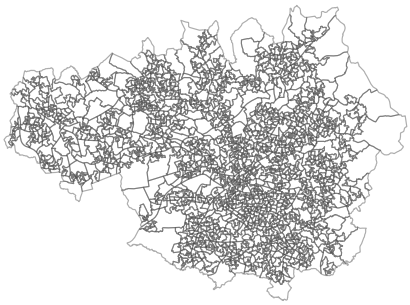
3.3.1 Choice of geographic units

Normally, GDCs involve the analysis of aggregated 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 Greater Manchester simply by plotting the geometry column of `st_LSOA`, which can be selected with the function `st_geometry()`. More sophisticated ways of visualising geographical data are available in R as we will see in forthcoming sections.

```
plot(st_geometry(sf_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 characteristics of the spatial units under study. In this case, we have prepared the file `manchester_land_cover_2011.gpkg` to contain some interesting land use data corresponding to each LSOA. The data frame `sf_LSOA` contains this data and we can visualise its first few lines by using the function `head()` and first four columns by subsetting the simple feature object:

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

```
Simple feature collection with 6 features and 4 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 368766.7 ymin: 411048.3 xmax: 372572.6 ymax: 414726.9
```

```
Projected CRS: OSGB36 / British National Grid
   LSOA11CD    LSOA11NM Sea.and.ocean..523...2012.
1 E01004766 Bolton 005A                           0
2 E01004767 Bolton 005B                           0
3 E01004768 Bolton 001A                           0
4 E01004769 Bolton 003A                           0
5 E01004770 Bolton 003B                           0
6 E01004771 Bolton 003C                           0
  Continuous.urban.fabric..111...2012.                          geom
1                                    0 MULTIPOLYGON (((371567 4119...
2                                    0 MULTIPOLYGON (((371807.5 41...
3                                    0 MULTIPOLYGON (((370197.2 41...
4                                    0 MULTIPOLYGON (((371924.3 41...
5                                    0 MULTIPOLYGON (((372572.6 41...
6                                    0 MULTIPOLYGON (((371554.4 41...
```

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 `sf_LSOA`

```
names(sf_LSOA)
```

```
 [1] "LSOA11CD"
 [2] "LSOA11NM"
 [3] "Sea.and.ocean..523...2012."
 [4] "Continuous.urban.fabric..111...2012."
 [5] "Discontinuous.urban.fabric..112...2012."
 [6] "Industrial.or.commercial.units..121...2012."
 [7] "Sport.and.leisure.facilities..142...2012."
 [8] "Non.irrigated.arable.land..211...2012."
 [9] "Mineral.extraction.sites..131...2012."
[10] "Pastures..231...2012."
[11] "Natural.grasslands..321...2012."
[12] "Moors.and.heathland..322...2012."
[13] "Peat.bogs..412...2012."
[14] "Transitional.woodland.shrub..324...2012."
[15] "Water.bodies..512...2012."
[16] "Sparsely.vegetated.areas..333...2012."
[17] "Coniferous.forest..312...2012."
[18] "Mixed.forest..313...2012."
[19] "Broad.leaved.forest..311...2012."
[20] "Construction.sites..133...2012."
```

```
[21] "Green.urban.areas..141...2012."
[22] "Port.areas..123...2012."
[23] "Land.principally.occupied.by.agriculture..with.significant.areas.of.natural.vegetation
[24] "Water.courses..511...2012."
[25] "Road.and.rail.networks.and.associated.land..122...2012."
[26] "Dump.sites..132...2012."
[27] "Airports..124...2012."
[28] "Intertidal.flats..423...2012."
[29] "Estuaries..522...2012."
[30] "Beaches..dunes..sands..331...2012."
[31] "Inland.marshes..411...2012."
[32] "Salt.marshes..421...2012."
[33] "Complex.cultivation.patterns..242...2012."
[34] "Coastal.lagoons..521...2012."
[35] "Bare.rocks..332...2012."
[36] "Fruit.trees.and.berry.plantations..222...2012."
[37] "Burnt.areas..334...2012."
[38] "Agro.forestry.areas..244...2012."
[39] "LSOA11NMW"
[40] "BNG_E"
[41] "BNG_N"
[42] "LONG"
[43] "LAT"
[44] "GlobalID"
[45] "geom"
```

We can explore the summary statistics for each of the variables with the `summary()` function applied on the variable of interest. For example, to obtain the summary statistics for the proportion of land dedicated to discontinuous urban fabric in 20212 `Discontinuous.urban.fabric..112...2012`, we can run the code below:

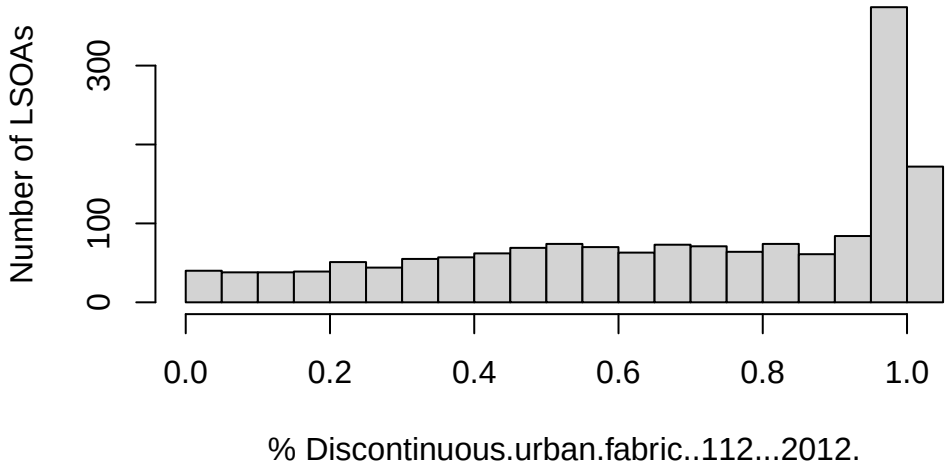
```
summary(sf_LSOA$Discontinuous.urban.fabric..112...2012.)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.4445  0.7474  0.6823  0.9958  1.0000
```

This tells us that the mean or average proportion of land dedicated to discontinuous urban fabric in LSOAs within Greater Manchester is 0.6823. It also tells us that 1 is the proportion of land dedicated to discontinuous urban fabric in the LSOA with the maximum proportion of land dedicated to this use.

To visualise the whole distribution of the variable `Discontinuous.urban.fabric..112...2012.`, we can plot a histogram:

```
hist(sf_LSOA$Discontinuous.urban.fabric..112...2012., breaks=15, xlab="% Discontinuous.urb
```

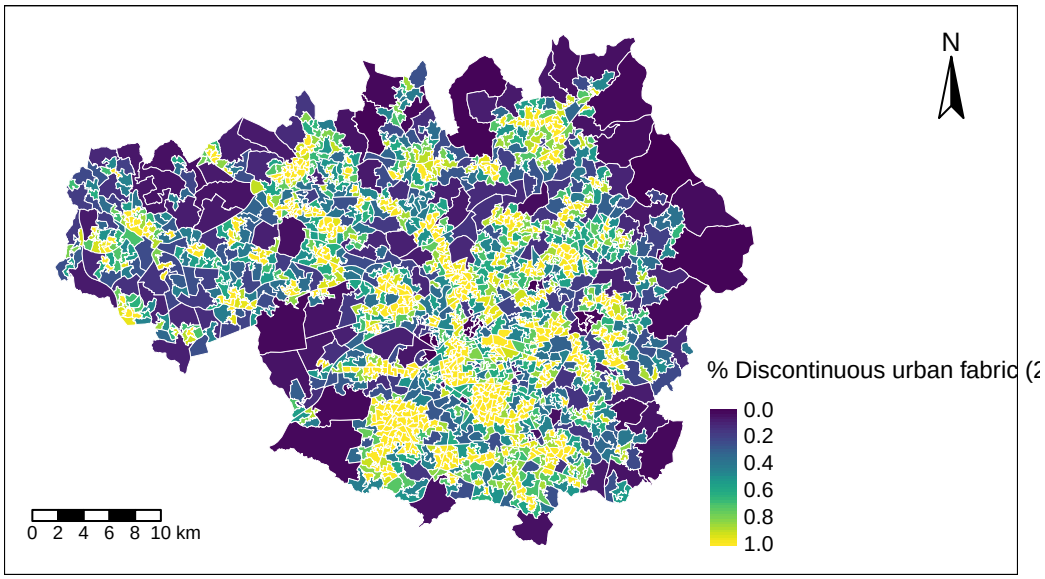


The histogram reveals that many LSOAs have a high proportion of discontinuous urban fabric, but there are a few with more than 50% of their land dedicated to this use.

Now the question is whether the LSOAs with similar proportions of discontinuous urban fabric are also spatially close. To find out, we need to map the data. We can do this by using the `tmap` library functions. We observe that, LSOAs in the South of Greater Manchester tend to have high proportions of discontinuous urban fabric, while LSOAs in the peripheries have a low proportion of this type of land use.

```
legend_title = expression("% Discontinuous urban fabric (2012)")
map_discountinuous = tm_shape(sf_LSOA) +
  tm_fill(col = "Discontinuous.urban.fabric..112...2012.", title = legend_title, text.size
  tm_layout(legend.position = c(0.69, 0.02), legend.title.size=0.9, inner.margins=c(0.05,
  tm_borders(col = "white", lwd = .01)  + # add borders
  tm_compass(type = "arrow", position = c("right", "top") , size = 2) + # add compass
  tm_scale_bar(breaks = c(0,2,4,6,8,10), text.size = 0.8, position =  c("left", "bottom"))
```

```
map_discountinuous
```



3.4 Standardisation

3.4.1 Across geographic units

LSOAs have been designed to have similar population sizes, so their area fluctuates. If the area of an LSOA is bigger or smaller, this can affect the figures corresponding to land use (e.g. presumably, the larger the total area, the hectares will be dedicated to a certain use).

To counter the effect of varying sizes across geographic units, we always need to standardise the data so it is given as a proportion or a percentage. This has already been done in the dataset `manchester_land_cover_2011.gpkg`, however, if you were to create your own dataset, you need to take this into account. To compute the right percentages, it is important to consider the right denominator. For example, if we are computing the percentage of people over the age of 65 in a given geographic unit, we can divide the number of people over 65 by the total population in that geographic unit, then multiply by 100. However, if we are computing the percentage of single-person households, we need to divide the number of single-person households by the total number of households (and not by the total population), then multiply by 100.

3.4.2 Variable standardisation

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

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

```
# creates a new data frame based on sf_LSOA
df_std <- sf_LSOA %>% st_drop_geometry()
# extracts column names from df_std
colnames_df_std <- colnames(df_std)

# loops columns from position 3 : the last column
for(i in 3: 38){
df_std[[c(colnames_df_std[i])]] <- scale(as.numeric(df_std[, colnames_df_std[i]]))
}
```

After the standardisation, some columns in `df_std` contain NaN. For simplicity, we will remove these columns from the resulting dataframe as below, and we will redefine the variable containing the column names so that it only stores the column names corresponding to landuse variables:

```
df_std <- df_std %>% select_if(~ !any(is.na(.)))
colnames_df_std <- colnames(df_std)[3:(ncol(df_std)-6)]
```

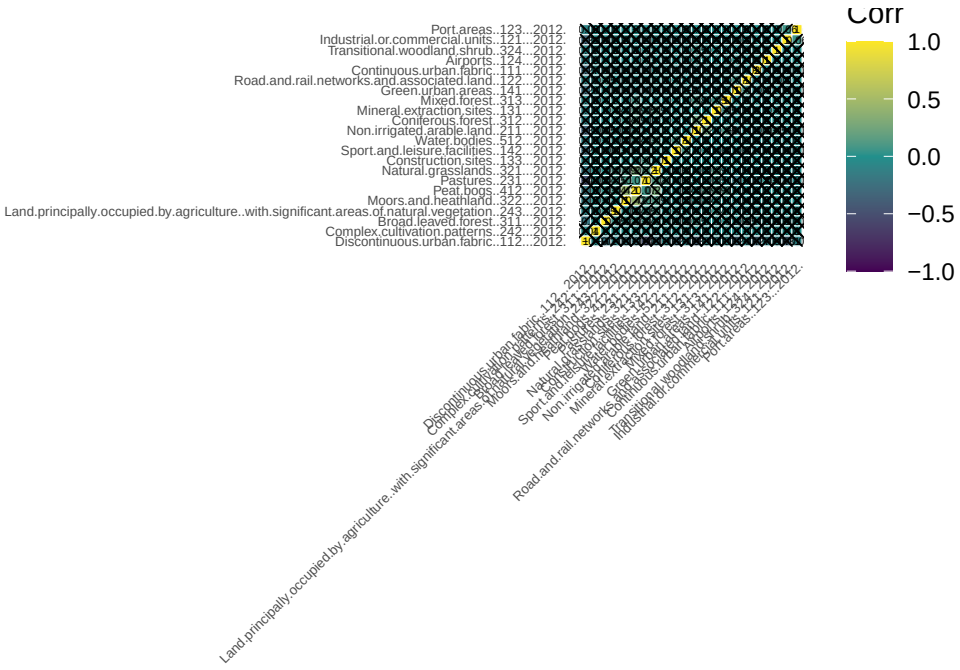
3.5 Checking for variable association

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 0.9, therefore, p-values smaller than 0.1 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 0.1). 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)], method = "pearson")
# Matrix of p-values
corr_pmat <- cor_pmat(df_std[,c(colnames_df_std)], method = "pearson", conf.level = 0.95)
# Barring the no significant coefficient
ggcorrplot(corr_mat, tl.cex=5, hc.order = TRUE, outline.color = "white", p.mat = corr_pmat
```



Among the statistically significant values in the correlation matrix, we can see that none of the variables display a strong correlation (i.e. close to -1 or 1), except of course, each variable with itself (as shown in the diagonal cells of the correlogram). In fact, many of the measured correlations are not significant. Therefore, we do not need to remove any variables from the analysis. However, if we had found a strong correlation between a pair of variables which is statistically significant, we could remove one of the two variables, since they would be both capturing very similar information.

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(df_std[,c(colnames_df_std)], centers=4, nstart=20, iter.max = 1000)
```

3.6.2 Number of clusters

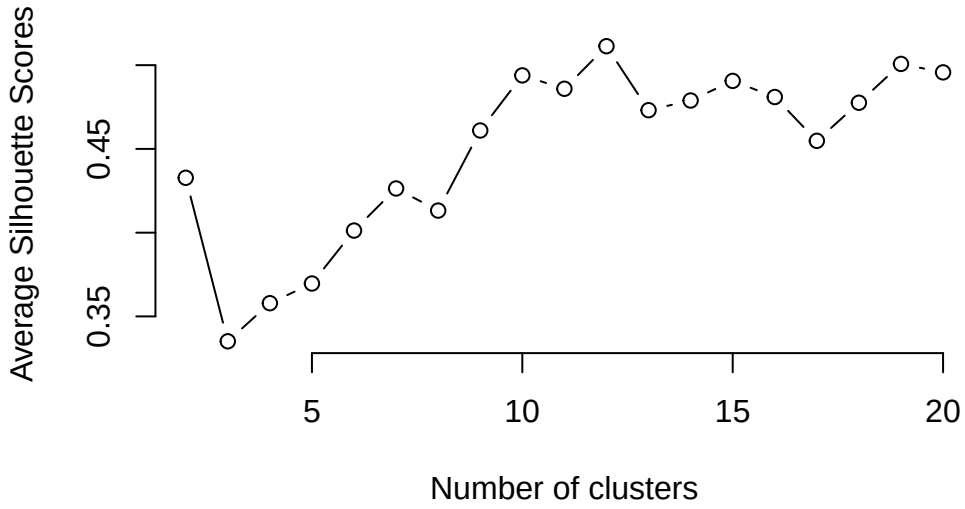
As mentioned above, the number of clusters k is a parameter of the algorithm that has to be specified by the user. Ultimately, there is no right or wrong answer to the question 'what is the optimum number of clusters?'. Deciding the value of k in the k-means algorithm can be a somewhat subjective process 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 20. The optimum number of clusters is given by the value of k at which the average silhouette is maximised.

```r
# Specify a random number generator seed for reproducibility
set.seed(123)

silhouette_score <- function(k){
  km <- kmeans(df_std[,c(colnames_df_std)], centers = k, nstart=5, iter.max = 1000)
  ss <- silhouette(km$cluster, dist(df_std[,c(colnames_df_std)]))
  mean(ss[, 3])
}
k <- 2:20
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 12. Note, this number might be different when you run the programme since the clustering algorithm involves some random steps, unless you set the seed to 123 as we did above. A number of clusters of k=12 is too high for our results to be insightful and interpretable. Therefore, to keep our interpretations simpler, we will pick a lower number of clusters to help us undertand our data better, such as 5.

```r
# Specify a random number generator seed for reproducibility
set.seed(123)
Km <- kmeans(df_std[,c(colnames_df_std)], 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 land use characteristics. We can include to our dataset the cluster where each geographical unit belongs to:

```
df_std$cluster <- Km$cluster
```

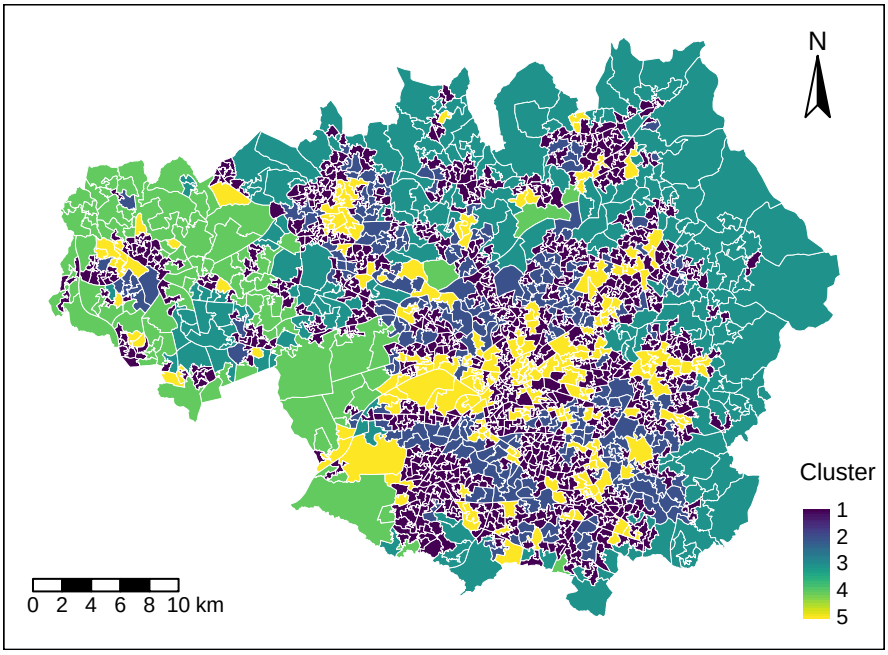
To map the results from clustering, we add the spatial inforamtion which we recover from `sf_LSOA`.

```
# Get the geometry column from the sf object
geometry_column <- st_geometry(sf_LSOA)
```

```
# Set the geometry column in the dataframe which becomes a simple feature object
sf_std <- st_set_geometry(df_std, geometry_column)
```

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

```
map_cluster = tm_shape(sf_std) +
  tm_fill(col = "cluster", title = "Cluster", palette = viridis(256), style = "cont") + #
  tm_borders(col = "white", lwd = .01)  + # add borders
  tm_layout(legend.position = c(0.9, 0.02), legend.title.size=0.9, inner.margins=c(0.05, 0
  tm_compass(type = "arrow", position = c("right", "top") , size = 2) + # add compass
  tm_scale_bar(breaks = c(0,2,4,6,8,10), text.size = 0.8, position =  c("left", "bottom"))

map_cluster
```



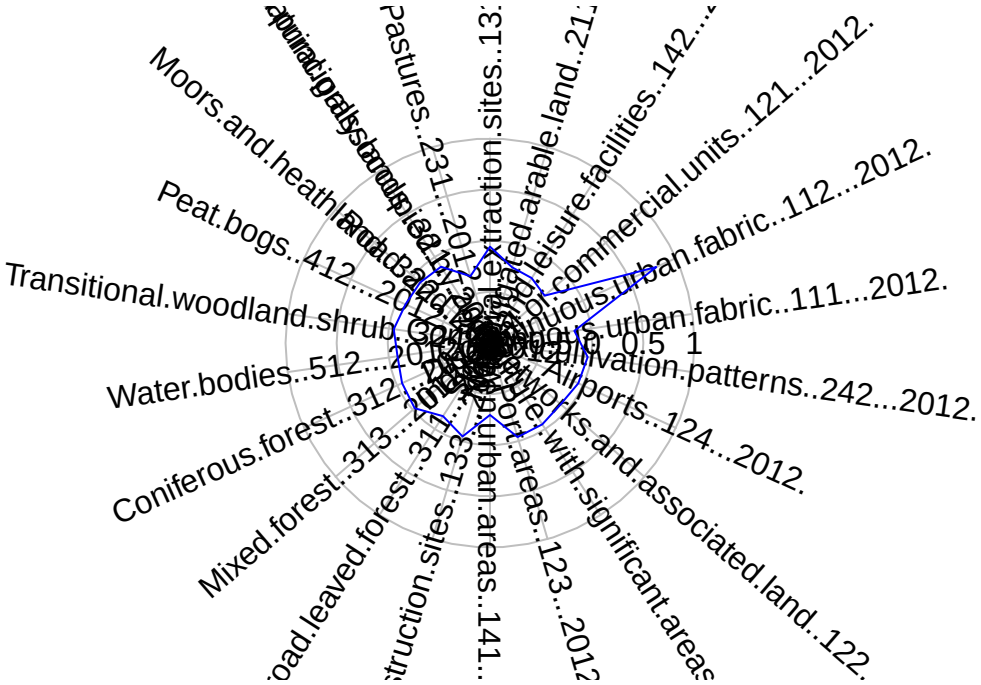
Note: sometimes, the number of items in a cluster may be very small. In that case, you may want to merge two clusterx to make the number of items in each group more homogeneous or perhaps change k in the k-means algorithm.

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 LSOAs with similar land use characteristics are 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 of the cluster where they belong based on the variables used in the classification. The data used in the clustering process was Z-score standardised, 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=1, mar=c(3,3,3,3))
```

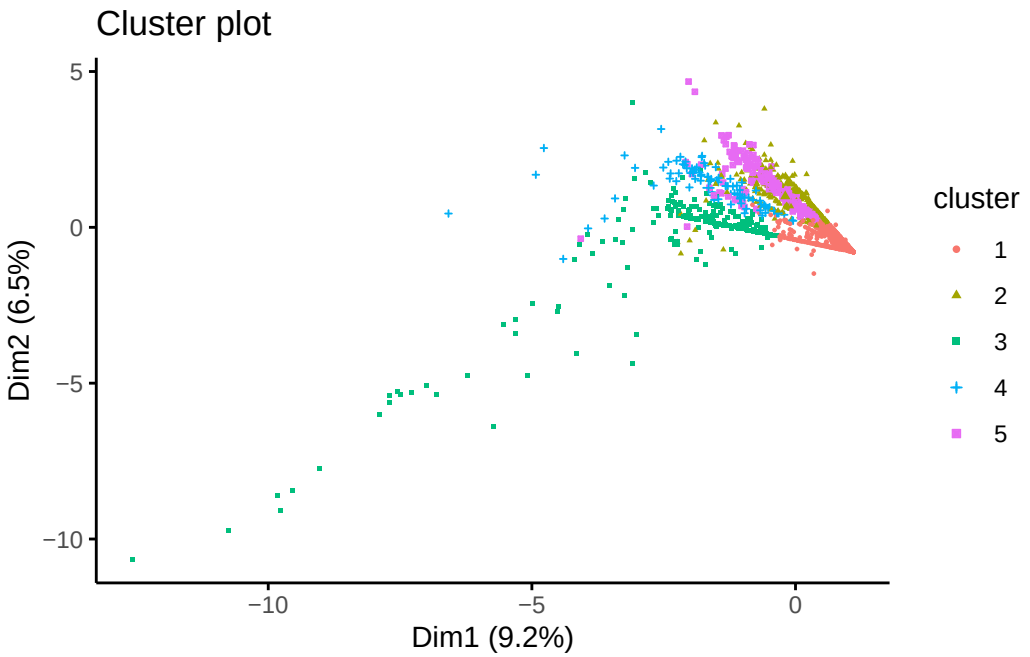


3.7.3 Testing

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

In the plot below, we can see the first and second principal components in the x and y axes respectively, with the axis label indicating the amount of variability that these components are able to explain. To create the plot, we use the `fviz_cluster()` function from the factoextra library.

```
fviz_cluster(Km, data = df_std[,c(colnames_df_std)], geom = "point", ellipse = F, pointsiz
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 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 dataset that we used for Chapter 3, but for Greater London:

```
sf_LSOA <- st_read("./data/geodemographics/london_land_cover_2011.gpkg")
```

```
Reading layer `london_land_cover_2011' from data source
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202324/r4ps/data/geodemographi
  using driver `GPKG'
Simple feature collection with 4835 features and 44 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
```

Prepare your data for a geodemographic classification (GDC). To do this, start by standardising the land use variables. Then, check for variable association using a correlation matrix. Discard any variables if necessary. Now you should be ready to group the data into clusters using the k-means algorithm. Report the optimal number of clusters based on the average silhouette score method. Select the number of clusters for a GDC with k-means according to this method or otherwise. Join the resulting dataset with the LSOA boundary data. 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.

```r
# 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. WoldPop produces these gridded datasets using a top-down (i.e. dissagregating 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 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//Rtmps5wYLq/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//Rtmps5wYLq/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 downloa
481  Estimated total number of people per grid-cell 2001 The dataset is available to downloa
730  Estimated total number of people per grid-cell 2002 The dataset is available to downloa
979  Estimated total number of people per grid-cell 2003 The dataset is available to downloa
1228 Estimated total number of people per grid-cell 2004 The dataset is available to downloa
1477 Estimated total number of people per grid-cell 2005 The dataset is available to downloa
```

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
```

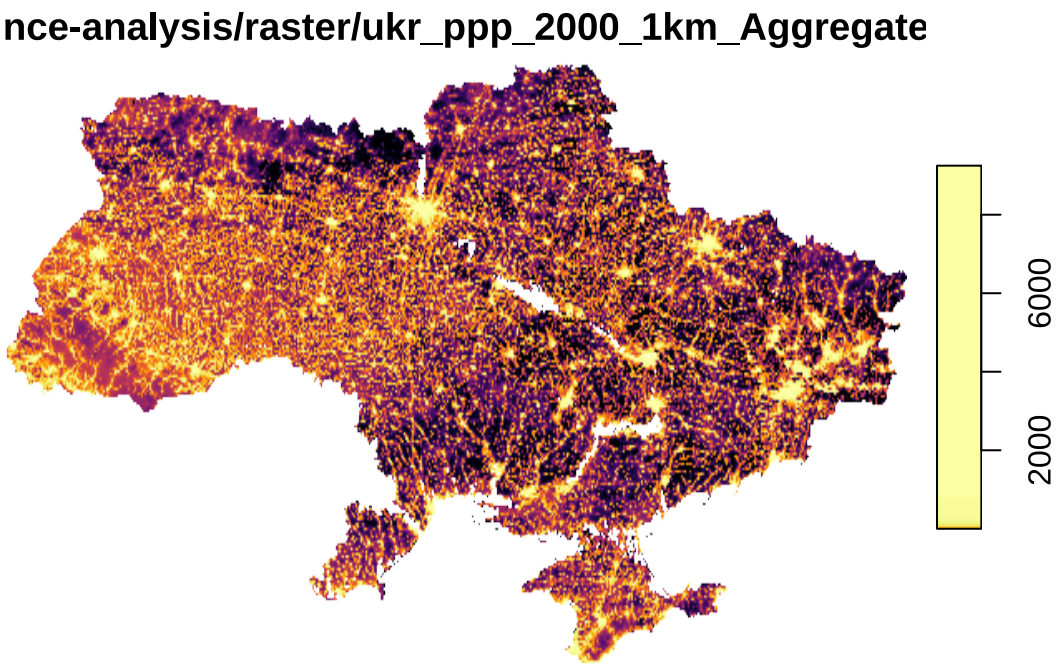
```
./data/sequence-analysis/raster/ukr_ppp_2000_1km_Aggregated_UNadj.tif
./data/sequence-analysis/raster/ukr_ppp_2001_1km_Aggregated_UNadj.tif
./data/sequence-analysis/raster/ukr_ppp_2002_1km_Aggregated_UNadj.tif
./data/sequence-analysis/raster/ukr_ppp_2003_1km_Aggregated_UNadj.tif
./data/sequence-analysis/raster/ukr_ppp_2004_1km_Aggregated_UNadj.tif
./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
```



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/envs418/202324/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                      ENGTYPE_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 proccess 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
 64.732   8.316  73.220
```

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)
```

```r
# 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... 2000
2  UKR.1.1_1        Cherkas'ka  280917.39 POLYGON ((32.1715 49.43881,... 2000
3  UKR.1.2_1       Cherkas'kyi   89116.78 POLYGON ((32.03393 49.49881... 2000
4  UKR.1.3_1   Chornobaivs'kyi   50096.24 POLYGON ((32.17991 49.44486... 2000
5  UKR.1.4_1     Chyhyryns'kyi   36646.73 POLYGON ((32.26144 49.20893... 2000
6  UKR.1.5_1       Drabivs'kyi   42467.86 POLYGON ((32.41852 49.83724... 2000
7  UKR.1.6_1 Horodyshchens'kyi   49886.59 POLYGON ((31.56959 49.42509... 2000
8  UKR.1.7_1       Kamians'kyi   35587.28 POLYGON ((32.19797 49.20946... 2000
9  UKR.1.8_1         Kanivs'ka   14406.93 MULTIPOLYGON (((31.4459 49.... 2000
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.

```r
# 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")
```

```r
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

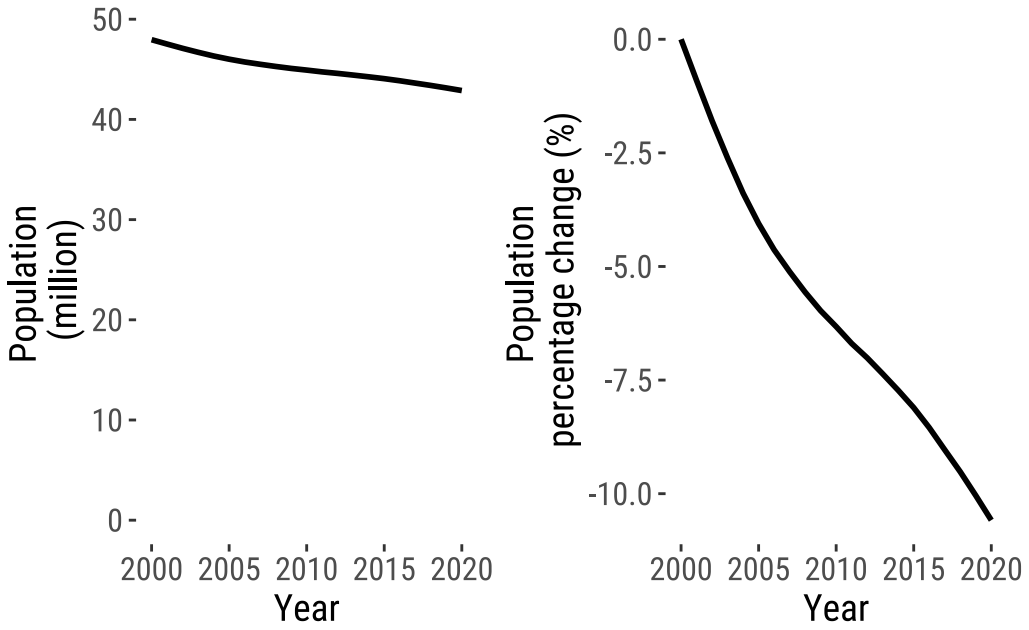
We first analyse national population trends. We want to know to what extent the population of Ukraine has declined over time over the last 20 years. An effective way to do this is to compute summary statistics and visualise the data. Below we look at year-to-year changes in

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 accelarators 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).

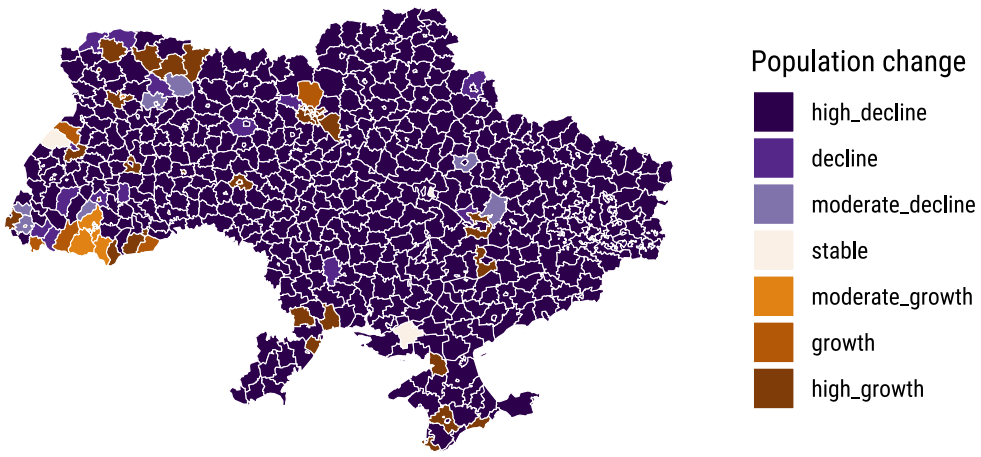
To examine the patterns of sun-national population losses, we compute two summary measures: (1) annual percentage change in population; and, (2) overall percentage change in population between 2000 and 2021. We start by looking the overall percentage change as it is easier to visualise. To this end, 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) * 10
  ave_pct_change_2000_21 = mean(pct_change, na.rm = TRUE)
) %>%
  ungroup()
```

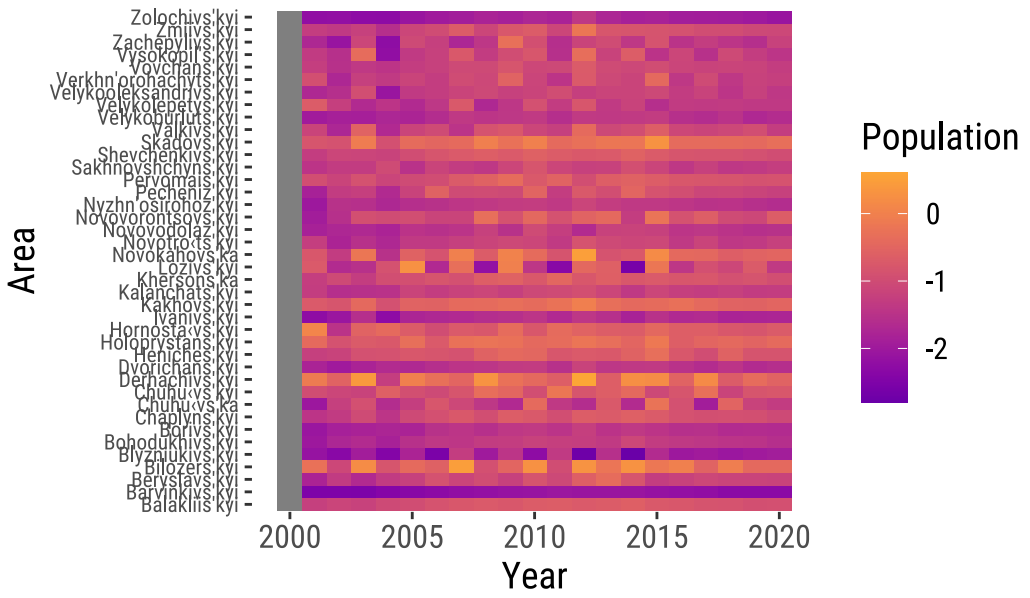
Let's map the overall percentage change in population between 2000 and 2020. We see a wide spread pattern of population decline across Ukraine. We observe a large spatial cluster of high population decline across the country with moderate population decline in some areas. Administrative areas containing 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 ~ 'modera
                           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 ~ 'moderat
                           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 Novovorontsovk'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

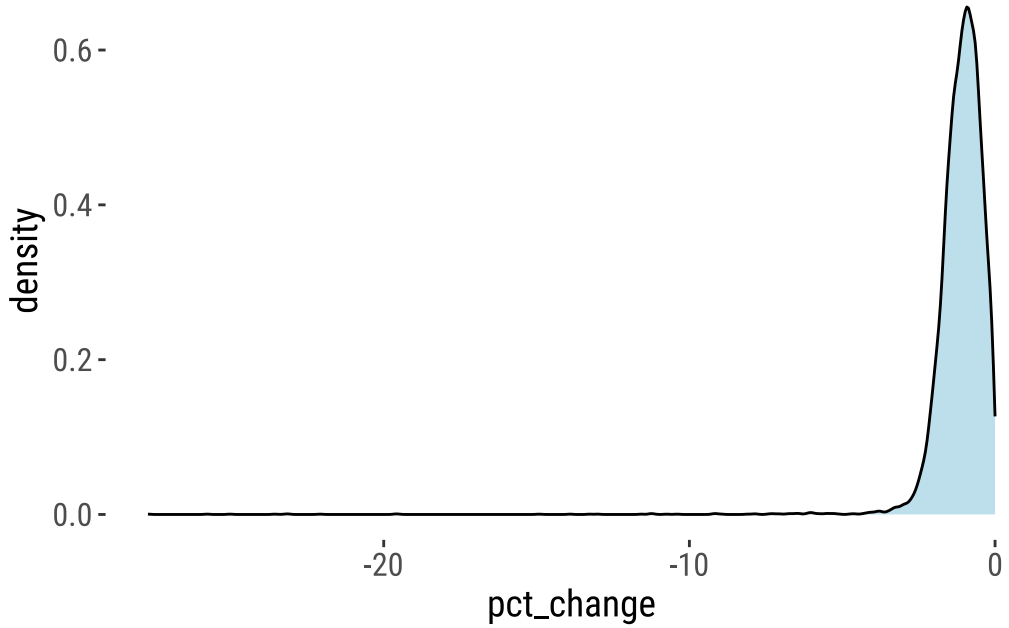
Sequence analysis requires longitudinal categorical data as an input. We therefore classify our population count data into distinct categorical categories, henceforth referred to as *states* of population change. We compute the annual percentage rate of population change for individual areas and use 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_tufte2()
```



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 is 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 $t1$ 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
   GID_2   `2001` `2002` `2003` `2004` `2005` `2006` `2007` `2008` `2009` `2010`
   <chr>   <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>
 1 UKR.1.~ decli~ moder~ decli~ decli~ decli~ moder~ moder~ moder~ moder~ moder~
 2 UKR.1.~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~
 3 UKR.1.~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~
 4 UKR.1.~ decli~ decli~ moder~ decli~ moder~ moder~ moder~ moder~ moder~ moder~
 5 UKR.1.~ decli~ decli~ moder~ decli~ moder~ moder~ moder~ moder~ moder~ moder~
 6 UKR.1.~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~
 7 UKR.1.~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~
 8 UKR.1.~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~ moder~
 9 UKR.1.~ decli~ decli~ decli~ decli~ decli~ decli~ decli~ decli~ moder~ decli~
10 UKR.1.~ moder~ stable moder~ moder~ stable moder~ moder~ stable decli~ stable
# i 561 more rows
# i 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.

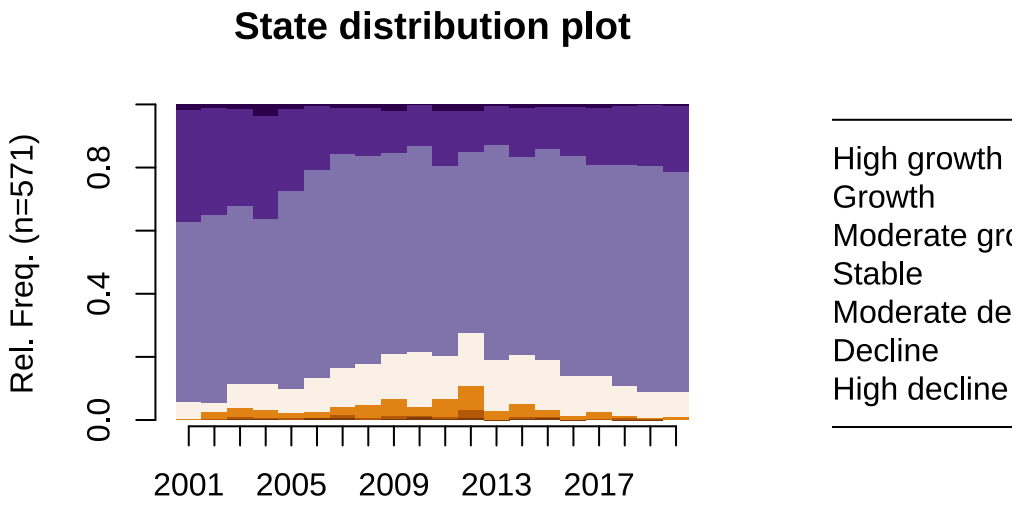
```
# alphabet
seq.alphab <- c("high_growth", "growth", "moderate_growth", "stable", "moderate_decline",
# labels
seq.lab <- c("High growth", "Growth", "Moderate growth", "Stable", "Moderate decline", "De
# define state sequence object
seq.cl <- seqdef(wide_population_loss_df,
                 2:21,
                 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 Ukrain 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 `seqtrate` . 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.trate <- seqtrate(seq.cl)
```

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

```
round(seq.trate, 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-meloids` . 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
```

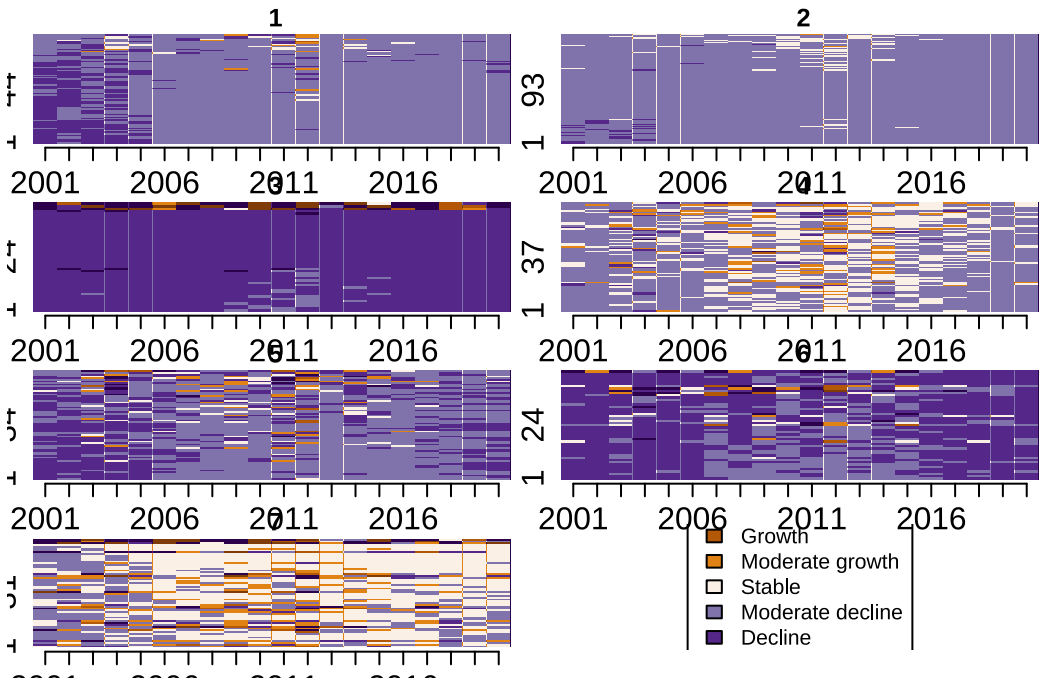
We rerun and save the results for a 7k cluster partition. If you inspect the resulting data frame, it provides an identifier for each cluster. Each individual area is attributed to a cluster. Next the question that we seek to answer is what sort of pattern do these clusters capture?

```
# 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)
```



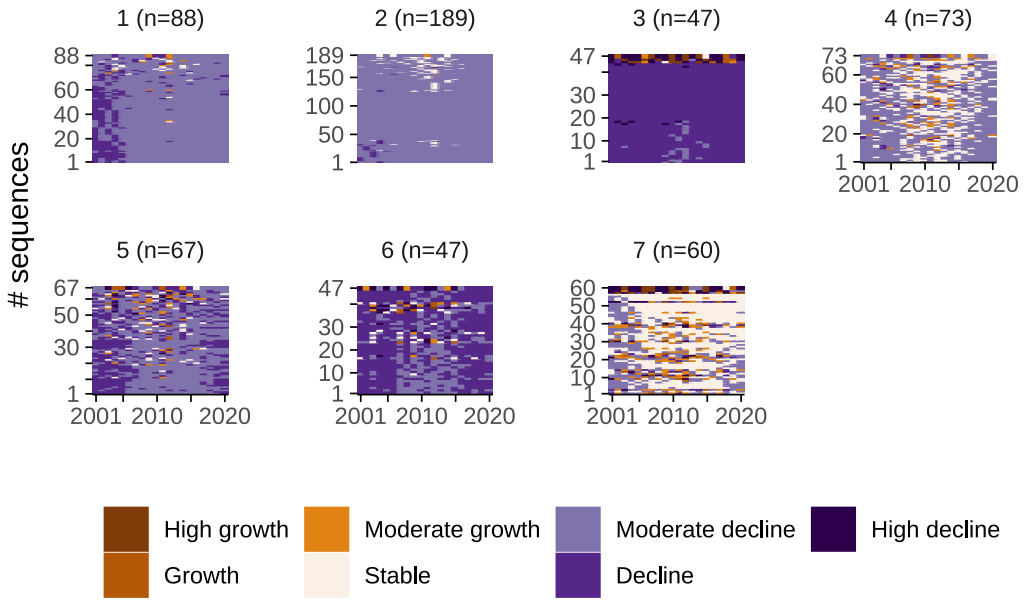
We therefore switch to the R library `ggseqplot` which enables visualisation of sequence data based on `ggplot` functionalities. This package may provide more flexibility if we are more familiar with `ggplot`.

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 and 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.
```

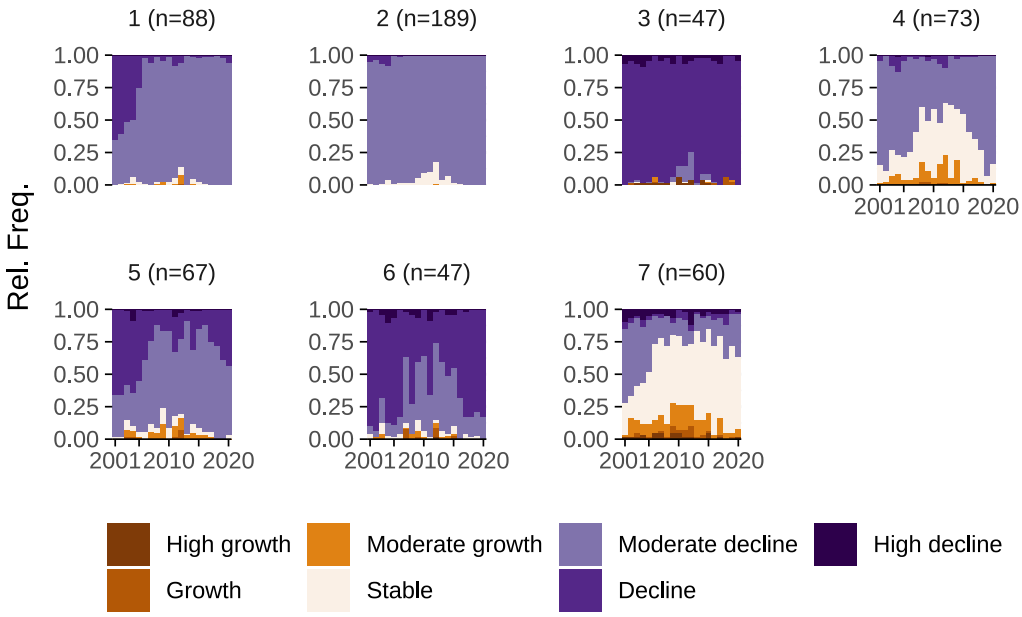


We can also get a better understanding of the resulting trajectories by analysing *state frequency plots*. They show the number of occurrences of a given state in individual years. These plots examine the data from a vertical perspective i.e. looking at individual years across areas, rather than at individual 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)
```

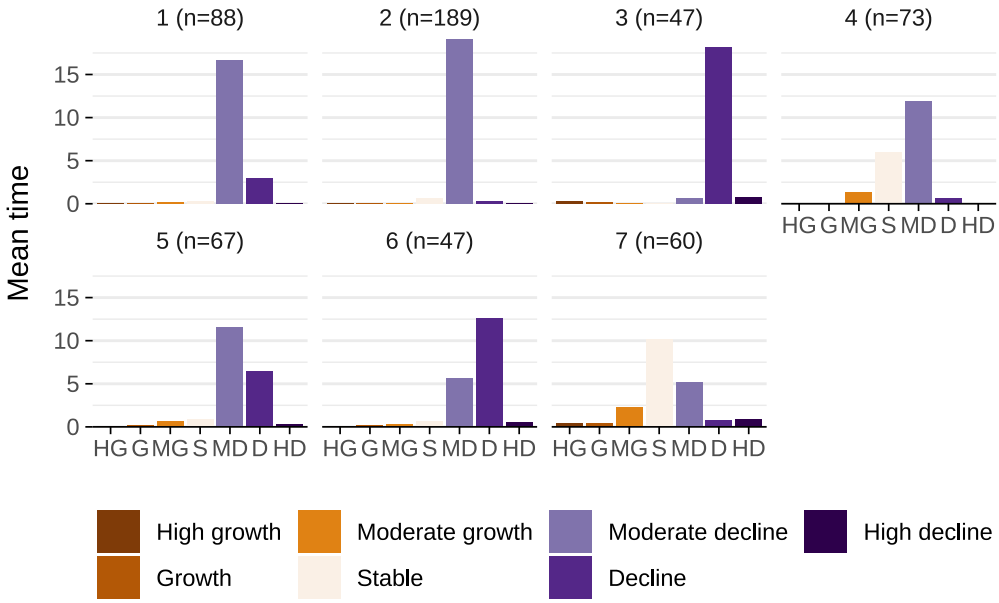
```
Scale for fill is already present.
Adding another scale for fill, which will replace the existing scale.
```



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
ggseqmtplot(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"
```

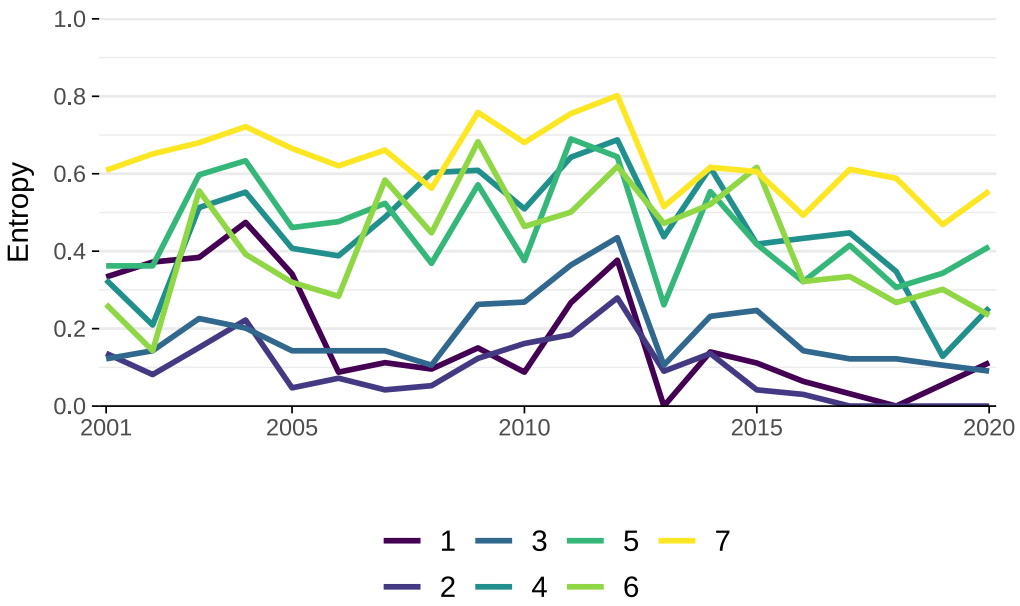
```
Scale for fill is already present.
Adding another scale for fill, which will replace the existing scale.
```



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              Geography   `Area (sq km)` Estimated population~1
   <chr>     <chr>             <chr>                <dbl>                  <dbl>
 1 K02000001 UNITED KINGDOM    Country            242741.                67081234
 2 K03000001 GREAT BRITAIN     Country            228948.                65185724
 3 K04000001 ENGLAND AND WALES Country            151047.                59719724
 4 E92000001 ENGLAND           Country            130310.                56550138
 5 E12000001 NORTH EAST        Region               8581.                 2680763
 6 E06000047 County Durham     Unitary Au~          2226.                  533149
```

```
 7 E06000005 Darlington        Unitary Au~          197.                  107402
 8 E06000001 Hartlepool        Unitary Au~           93.7                  93836
 9 E06000002 Middlesbrough     Unitary Au~           53.9                 141285
10 E06000057 Northumberland    Unitary Au~         5020.                  323820
# i 410 more rows
# i abbreviated name: 1: `Estimated population mid-2020`
# i 39 more variables: `2020 people per sq. km` <dbl>,
#   `Estimated Population mid-2019` <dbl>, `2019 people per sq. km` <dbl>,
#   `Estimated Population mid-2018` <dbl>, `2018 people per sq. km` <dbl>,
#   `Estimated Population mid-2017` <dbl>, `2017 people per sq. km` <dbl>,
#   `Estimated Population mid-2016` <dbl>, `2016 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
```

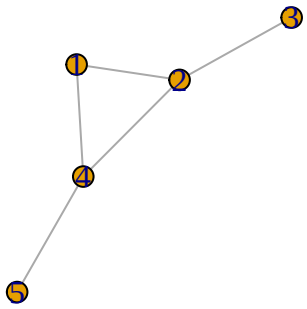
5.3 Creating networks

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

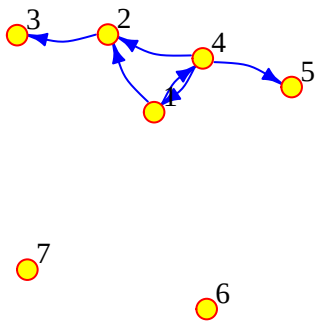
In order to create, manipulate and analyse networks in R, we will use the igraph package, which we imported in Section 5.2. We start by creating a very simple network with the code below. The network contains five nodes and five edges and it is undirected, so the edges do not have orientations. The nodes 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)
```



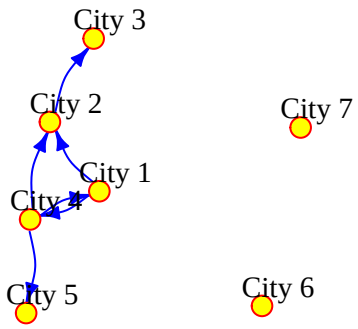
If the connections between the nodes of a network are non-reciprocal, the network is called directed. For example, this could correspond to a situation where there are people moving from city 1 to city 2, but nobody moving from city 2 to city 1. Note that in the code below we have not only added directions 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 d6bc2e4:
[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 d6bc2e4 (vertex names):
[1] City 1->City 2 City 2->City 3 City 1->City 4 City 4->City 1 City 4->City 2
[6] City 4->City 5
```

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

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

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

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

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

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

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

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

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

Finally, it is possible to add network attributes

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

5.4 Reading networks from data files

5.4.1 Preparing the data to create an igraph object

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

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

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

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

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

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

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

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

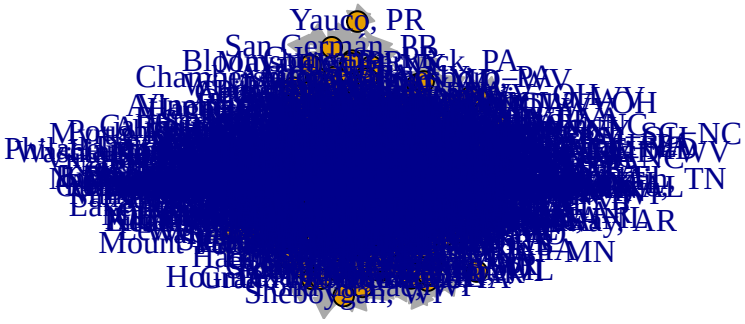
```
df_unique_origins <- df %>%
  distinct(MSA_Previous_Name_State) %>%
  rename(name = MSA_Previous_Name_State)

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_Previou
```

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

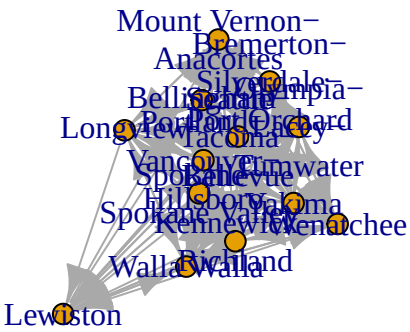
```
# Create a new data frame containing the columns for the origin, destination, and weight f
# Rename the columns to origin, destination, and weight respectively
df_sub_edges <- data.frame(df_sub$MSA_Previous_Name, df_sub$MSA_Current_Name, df_sub$Mover
  rename(origin = df_sub.MSA_Previous_Name, destination = df_sub.MSA_Current_Name, weight

# 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_N
# 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 node
# 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

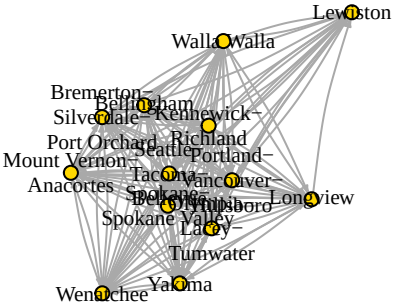
Let us start by generating the most basic visualisation of *g_sub*.

```
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*.

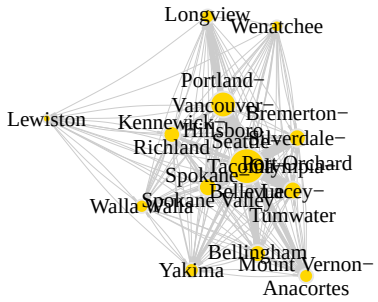
```
V(g_sub)$inflow <- rowSums(as.matrix(g_sub[]))
```

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/envs418/202324/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(grepl('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.

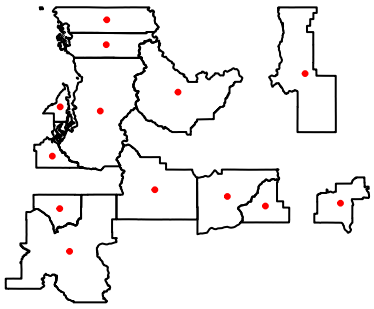
```
# Add longitude and latitude corresponding to centroid of each MSA polygon
msa_us$lon_centroid <- st_coordinates(st_centroid(msa_us$geometry))[,"X"]
msa_us$lat_centroid <- st_coordinates(st_centroid(msa_us$geometry))[,"Y"]
```

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(grepl('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)
```

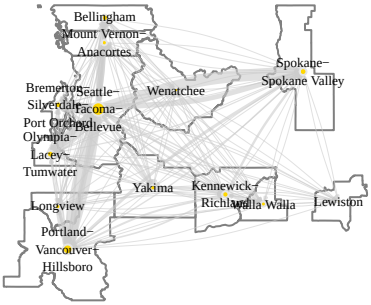


However, we still need to link this data to the network data that we obtained before. In order to incorporate the geographic information to the nodes of the migration subnetwork, we can join data from two data frames: *msa_sub*, which contains the geographic data, and *df_sub_nodes*, which contains the 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 centroi
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)
```

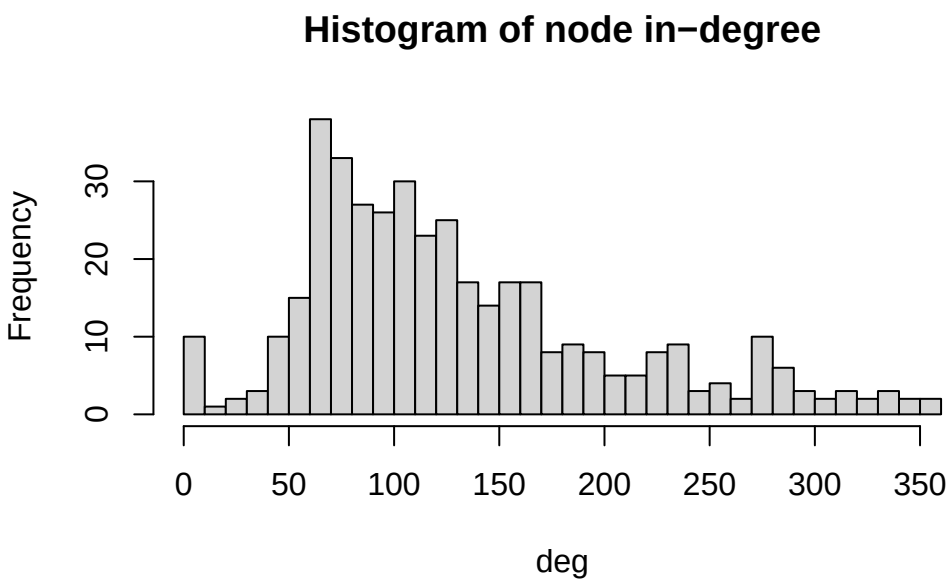
```
[1] 0.6067259
```

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-degr
deg <- degree(g_US, v=V(g_US), mode="in")

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



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

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

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

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

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

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

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

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

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

5.6.4 Distances

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 weigth at
output = "both") # outputs both path nodes and edges
```

```
$vpath
$vpath[[1]]
+ 2/402 vertices, named, from a7107c3:
[1] New York-Newark-Jersey City, NY-NJ-PA Los Angeles-Long Beach-Anaheim, CA


$epath
$epath[[1]]
+ 1/52930 edge from a7107c3 (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
```

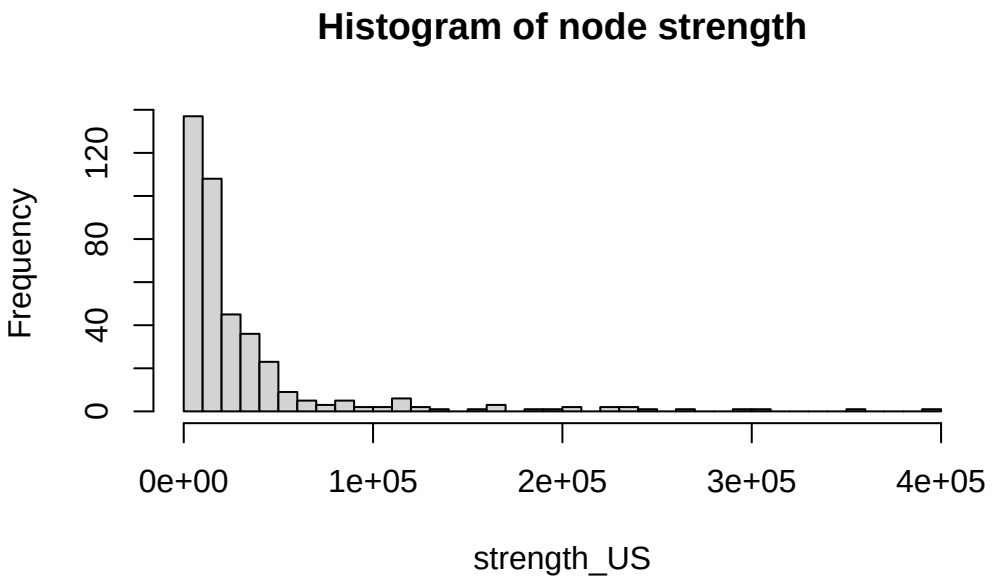
5.6.5 Centrality

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 w
)

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



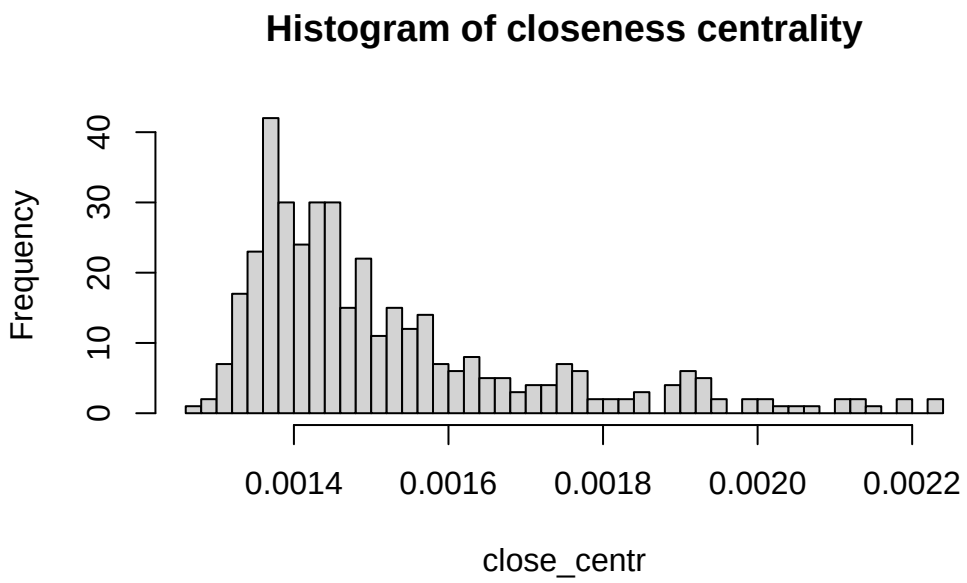
We can check which is the city with the maximum strength:

```
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"
```

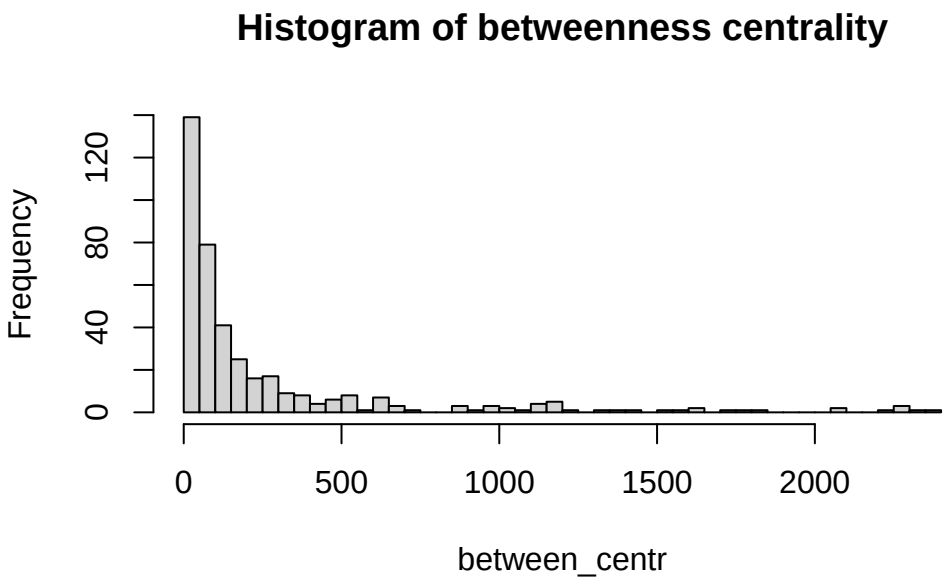
We will look at another two important centrality metrics that are based on the structure of the network. Firstly, **closeness centrality** which is a measure of the length of the shortest path between a node and all the other nodes. For a given node, it is computed as the inverse of the average shortest 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.

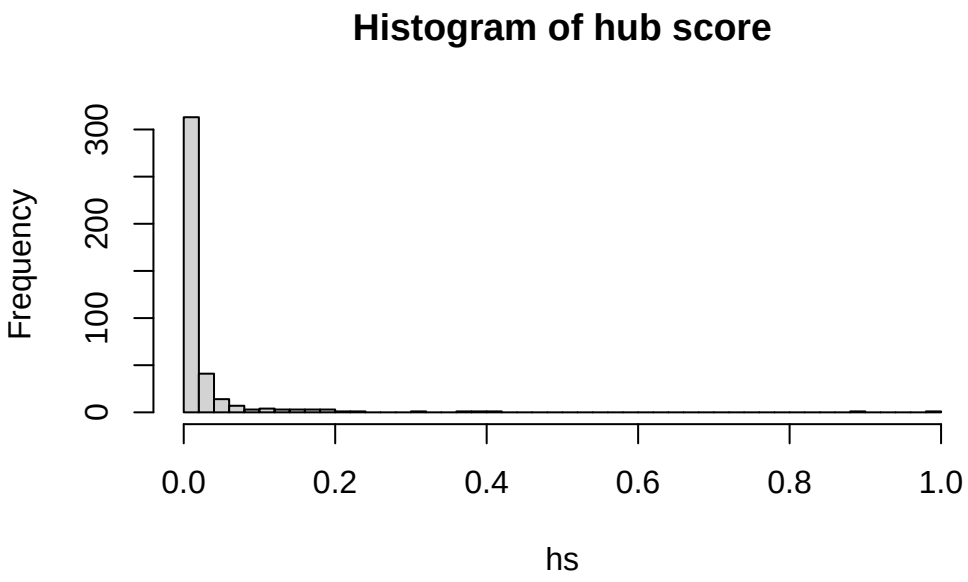
```
between_centr <- betweenness(g_US, v = V(g_US), directed = TRUE, weights = NA)
hist(between_centr, breaks = 50, main="Histogram of betweenness centrality")
```



5.6.6 Hubs and authorities

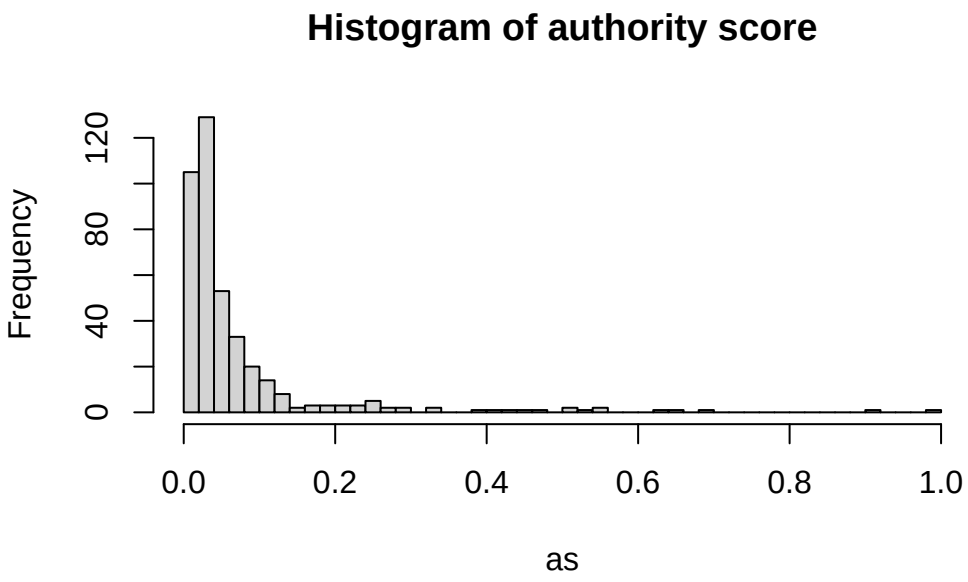
We call hubs or authorities those nodes with a higher-than-average degree. Normally, the name hub is reserved to nodes with high out-degree whereas authority is reserved to nodes with high in-degree. An algorithm to detect hubs and authorities was developed by Jon Kleinberg, although it was initially 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")
```



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")
```



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/envs418/202324/r4ps/data/networks/Loca
  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

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

# 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 descendent 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: 224,611
Columns: 4
$ id         <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1~
$ created_at <dttm> 2019-12-01 00:00:18, 2019-12-01 00:00:23, 2019-12-01 00:00~
$ status_id  <dbl> 1.200928e+18, 1.200928e+18, 1.200928e+18, 1.200928e+18, 1.2~
$ text       <chr> "Another example @anonymous are the TRUE protectors of sanc~
```

6.2.1 Text data structures

Different approaches to storing and manipulating text data exist. General formats include:

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 metadata 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.

TidyText: This is a novel approach developed by Sielge and Robinson (2022). The tidytext approach defines a representation in the form of a table with one-token-per-row. A token is a meaningful unit of text for analysis, such as a word, a sentence or a tweet. Tokenisation is the process of splitting text into tokens. Tidy data has a specific structure. Each variable is a column. Each observation is a row and each type of observational unit is a table, The R `tidytext` package provides full functionality to implement the tidytext approach and manipulate text with the standard `tidyverse` ecosystem.

```
# 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: 129,554 x 2
   word             n
   <chr>        <int>
 1 anonymous   449611
 2 the         215293
 3 to          163434
 4 http        143377
 5 url_removed 136774
 6 and         112320
 7 of          106244
 8 in           93216
 9 a            92763
10 is           75237
# i 129,544 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[40:50])
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
```

Let's see one of those tweets:

```
tweets_df$text[48]
```

```
[1] "Job Lead! ---&gt; LIFE Program Assistant, International Institute of Los Angeles http:/
```

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[48]`).

```
gsub("---&gt", "", tweets_df$text[48])
```

```
[1] "Job Lead! ; LIFE Program Assistant, International Institute of Los Angeles http://url_re
```

`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: 1,296 x 2
   created_at          ngram
```

```
   <dttm>              <chr>
 1 2019-12-01 00:00:18 another example
 2 2019-12-01 00:00:18 example anonymous
 3 2019-12-01 00:00:18 anonymous are
 4 2019-12-01 00:00:18 are the
 5 2019-12-01 00:00:18 the true
 6 2019-12-01 00:00:18 true protectors
 7 2019-12-01 00:00:18 protectors of
 8 2019-12-01 00:00:18 of sanctuary
 9 2019-12-01 00:00:18 sanctuary communities
10 2019-12-01 00:00:18 communities when
# i 1,286 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 with `by = join_by(word)`
```

Let's see what are the top words now:

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

```
# A tibble: 9 x 2
  word            n
  <chr>       <int>
1 trust           2
2 anonymous       1
3 citizens        1
4 communities     1
5 ice             1
6 police          1
7 protectors      1
8 sanctuary       1
9 true            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: 282 x 2
   created_at          word
   <dttm>              <chr>
 1 2019-12-01 00:00:18 Another
 2 2019-12-01 00:00:18 example
 3 2019-12-01 00:00:18 anonymous
 4 2019-12-01 00:00:18 are
 5 2019-12-01 00:00:18 the
 6 2019-12-01 00:00:18 TRUE
 7 2019-12-01 00:00:18 protectors
 8 2019-12-01 00:00:18 of
 9 2019-12-01 00:00:18 sanctuary
10 2019-12-01 00:00:18 communities
# i 272 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"   "true"        "protectors"  "sanctuary"   "communities"
 [6] "citizens"    "trust"       "police"      "trust"       "ice"
[11] "politics"    "ahead"       "public"      "safety"      "http"
[16] "url_removed" "218th"       "illegal"     "immigrant"   "child"
```

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 2019-12-01 00:00:18 anonym
 2 2019-12-01 00:00:18 true
 3 2019-12-01 00:00:18 protector
 4 2019-12-01 00:00:18 sanctuari
```

```
 5 2019-12-01 00:00:18 communiti
 6 2019-12-01 00:00:18 citizen
 7 2019-12-01 00:00:18 trust
 8 2019-12-01 00:00:18 polic
 9 2019-12-01 00:00:18 trust
10 2019-12-01 00:00:18 ice
11 2019-12-01 00:00:18 polit
12 2019-12-01 00:00:18 ahead
13 2019-12-01 00:00:18 public
14 2019-12-01 00:00:18 safeti
15 2019-12-01 00:00:18 http
16 2019-12-01 00:00:18 url_remov
17 2019-12-01 00:00:23 218th
18 2019-12-01 00:00:23 illeg
19 2019-12-01 00:00:23 immigr
20 2019-12-01 00:00:23 child
```

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: 123,886 x 2
   word             n
   <chr>        <int>
 1 anonymous   449611
 2 http        143377
 3 url_removed 136774
 4 immigration  51480
 5 people       19523
 6 ice          19397
 7 immigrant    18898
 8 trump        17861
 9 virus        16162
```

```
10 illegals     14392
# i 123,876 more rows
```

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

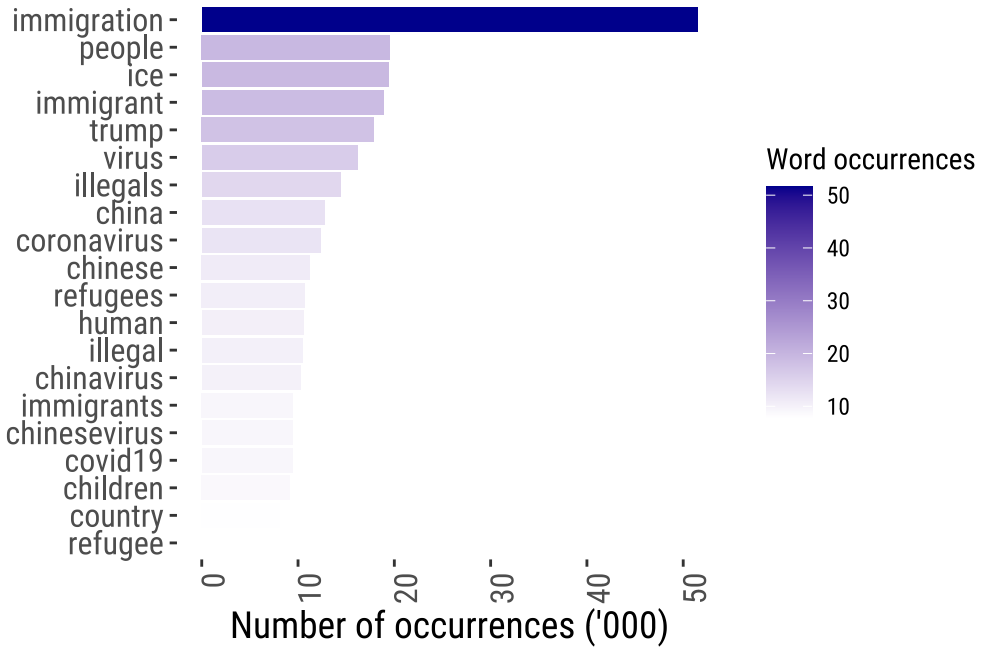
```
coding_words <- c("https|http|t.co|amp|anonymous|url_removed|http://url_removed")

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

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

```
# A tibble: 123,191 x 2
   word            n
   <chr>       <int>
 1 immigration 51480
 2 people      19523
 3 ice         19397
 4 immigrant   18898
 5 trump       17861
 6 virus       16162
 7 illegals    14392
 8 china       12750
 9 coronavirus 12318
10 chinese     11258
# i 123,181 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`, `bing`, `nrc` 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
# i 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
# i 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
# i 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 with `by = join_by(word)`
```

```
Warning in inner_join(., get_sentiments("nrc") %>% filter(sentiment %in% : Detected an unexpe
i Row 102 of `x` matches multiple rows in `y`.
i Row 812 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```
nrc_scores %>% head()
```

```
# A tibble: 6 x 5
  created_at          word      date       sentiment method
  <dttm>              <chr>     <date>     <chr>     <chr>
1 2019-12-01 00:00:18 true      2019-12-01 positive  nrc
2 2019-12-01 00:00:18 sanctuary 2019-12-01 positive  nrc
3 2019-12-01 00:00:18 police    2019-12-01 positive  nrc
4 2019-12-01 00:00:18 ahead     2019-12-01 positive  nrc
5 2019-12-01 00:00:18 public    2019-12-01 positive  nrc
6 2019-12-01 00:00:23 illegal   2019-12-01 negative  nrc
```

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

```
Joining with `by = join_by(word)`
```

```
Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relatio
i Row 106476 of `x` matches multiple rows in `y`.
i Row 6177 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```r
bing_scores %>% head()
```

```
# A tibble: 6 x 5
  created_at          word    date       sentiment method
  <dttm>              <chr>   <date>     <chr>     <chr>
1 2019-12-01 00:00:18 trust   2019-12-01 positive  bing
2 2019-12-01 00:00:18 trust   2019-12-01 positive  bing
3 2019-12-01 00:00:23 illegal 2019-12-01 negative  bing
4 2019-12-01 00:00:23 abuse   2019-12-01 negative  bing
5 2019-12-01 00:00:41 threats 2019-12-01 negative  bing
6 2019-12-01 00:00:41 protect 2019-12-01 positive  bing
```

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

```
Joining with `by = join_by(word)`
```

```r
afinn_scores %>% head()
```

```
# A tibble: 6 x 5
  created_at          word    date       value method
  <dttm>              <chr>   <date>     <dbl> <chr>
1 2019-12-01 00:00:18 true    2019-12-01     2 afinn
2 2019-12-01 00:00:18 trust   2019-12-01     1 afinn
```

```
3 2019-12-01 00:00:18 trust   2019-12-01     1 afinn
4 2019-12-01 00:00:18 safety  2019-12-01     1 afinn
5 2019-12-01 00:00:23 illegal 2019-12-01    -3 afinn
6 2019-12-01 00:00:23 arrests 2019-12-01    -2 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 with `by = join_by(word)`
```

```
Warning in inner_join(., get_sentiments("nrc") %>% filter(sentiment %in% : Detected an unexpe
i Row 102 of `x` matches multiple rows in `y`.
i Row 812 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
```

```
  "many-to-many"` to silence this warning.
```

```
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 with `by = join_by(word)`
```

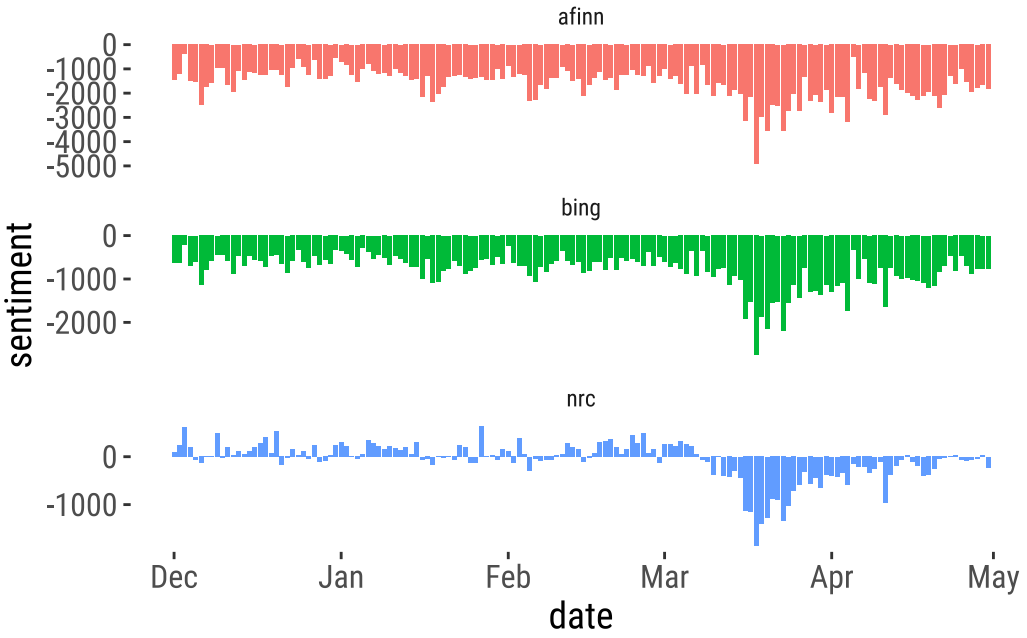
```
Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relati
i Row 106476 of `x` matches multiple rows in `y`.
i Row 6177 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```
afinn_scores <- tidy_tweets_df %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(date) %>%
  summarise(sentiment = sum(value)) %>%
  mutate(method = "afinn")
```

```
Joining with `by = join_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 there 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         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 <- vader_df(tweets_df$text)
```

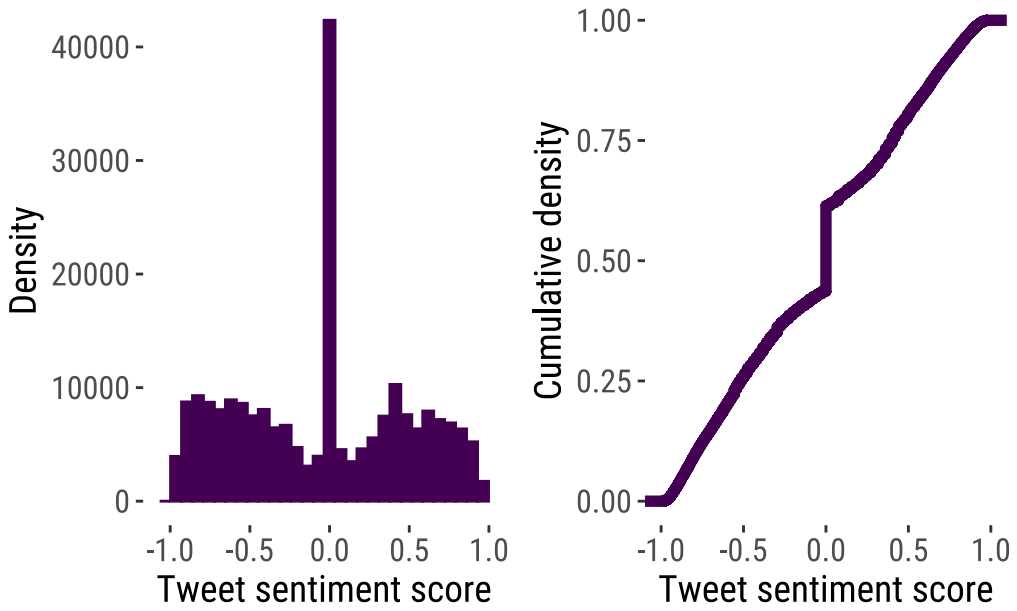
```
# combine vader scores,tweet ids and dates
tweets_scores_df <- cbind(tweets_df$id, tweets_df$created_at, tweet_vader_scores)

# 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
```

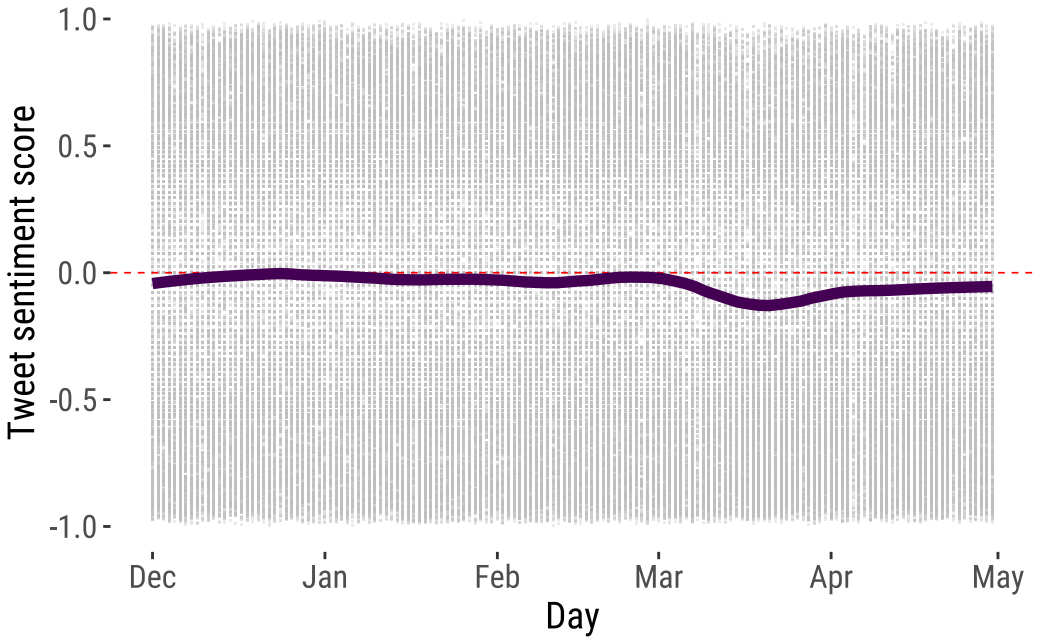


We produce two plots exploring the frequency and cumulative distribution of migration sentiment scores. The results indicate that a concentration around zero and also at both extremes i.e. below -0.5 and over 0.5, suggesting that migration is very polarising social issue.

Temporal evolution

We can also explore the temporal evolution of sentiment towards migration over time. The results indicate that migration sentiment remained slighly 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
```

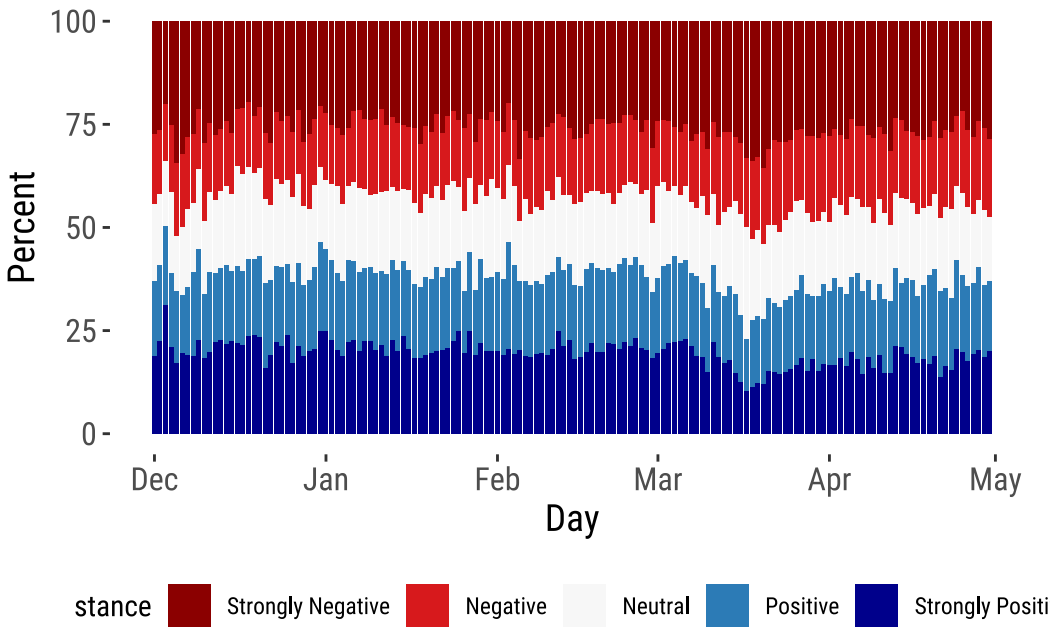


Composition

Analysing the coumpound 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 rethoric 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,
           )
         )
```

```r
# 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(com
  .[, 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", "
  labs(x= "Day",
       y = "Percent")

p4
```



> **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

Topic modelling is part of the larger topic of text data mining. Text mining is the process of transforming unstructured text into a structured format to identify meaningful patterns. In text mining, we often have collections of documents, such as news articles, blog posts, academic papers and much more. We often want to divide these documents into natural groups so that we can understand them separately. **Topic modeling** is a method for unsupervised classification of such documents, similar to clustering on numeric data, which finds natural groups of items - instead of counting them individually - even when we are not sure what we are looking for. Topic modeling is not the only method that does this– cluster analysis, latent semantic analysis, and other techniques have also been used to identify clustering within texts (Bail, 2020).

Topic models offer two significant advantages over simple forms of cluster analysis such as k-means clustering. Unlike k-means clustering, which assigns each document to only one cluster, topic models are mixture models that assign a probability to each document indicating its likelihood of belonging to a latent theme or topic.

Additionally, topic models use more advanced iterative Bayesian techniques to determine the probability of each document being associated with a particular theme or topic. Initially, documents are assigned a random probability of topic assignment, but the accuracy of these probabilities improves as more data is processed.

Topic Modelling has been used in population studies on various occasions to analyse demographic processes such as fertility and migration. Marshall (2013) for example, uses topic modelling to show the set of concepts relevant to the study of fertility was defined differently in France and Great Britain. Findings indicate that bith cultural and institutional differences were present in the research agendas around the understandings of fertility decline. This chapter will illustrate Topic Modelling with Reddit data.

Specifically it will investigate what reddit data can tell us about discussions around fertility and the pandemic's influence on fertility rates. Does the data shed any light on theories on the increase in fertility associated with rising wage inequality Bar et al. (2018)? Or on the negative impact of the pandemic on the fertility of women of prime childbearing age—30- to 34-year-olds?

This chapter is based on:

- The Topic modelling chapter in Silge, J. and Robinson, D, 2022. Text Mining with R: A Tidy Approach

- Bail, C. 2020's Topic Modelling chapter in *Text as DATA*. Computational Social Science

Latent Dirichlet Allocation

A widely used approach for creating a topic model is Latent Dirichlet Allocation (LDA). LDA considers

- Each document as a combination of topics. We imagine that each document may contain words from several topics in particular proportions. For instance, in a two-topic model we could say Document 1 is 80% about *migration* and 20% about *refugees*, while Document 2 is 40% about *migration* and 60% about *increased work-force*.
- Each topic as a blend of words. For example, we could imagine a two-topic model of a Twitter-feed, with one topic for "migration" and one for "refugees." The most common words in the migration topic might be "migrant", "origin", and "destination", while the refugee topic may be made up of words such as "armed conflict", "persecution", and "camp". Importantly, words can be shared between topics; a word like "destination" might appear in both equally.

This approach allows for documents to share content and overlap with one another, as opposed to being isolated into distinct groups. This mimics how natural language is typically used. Blei, Ng, and Jordan (2003) describe LDA's in detail.

For more on LDA's see Bail's chapter on Topic Modelling.

7.1 Dependencies

As shown in the Figure below by we can use tidy text principles to approach topic modeling with the same set of tidy tools used for other data analysis in R. In this chapter, we'll learn to work with LDA objects from the `topicmodels` package, tidying such models so that they can be analysed with the help of `ggplot2` and `dplyr`.

We use the libraries below.

```
#Topic models package that allows tidying such models with ggplot2 and dplyr
library(topicmodels)
#A framework for text mining applications within R.
library(tm)
#The Life-Changing Magic of Tidying Text
library(tidytext)
# Snowball Stemmers Based on the C 'libstemmer' UTF-8 Library
library(SnowballC)
# Data manipulation
library(tidyverse)
```

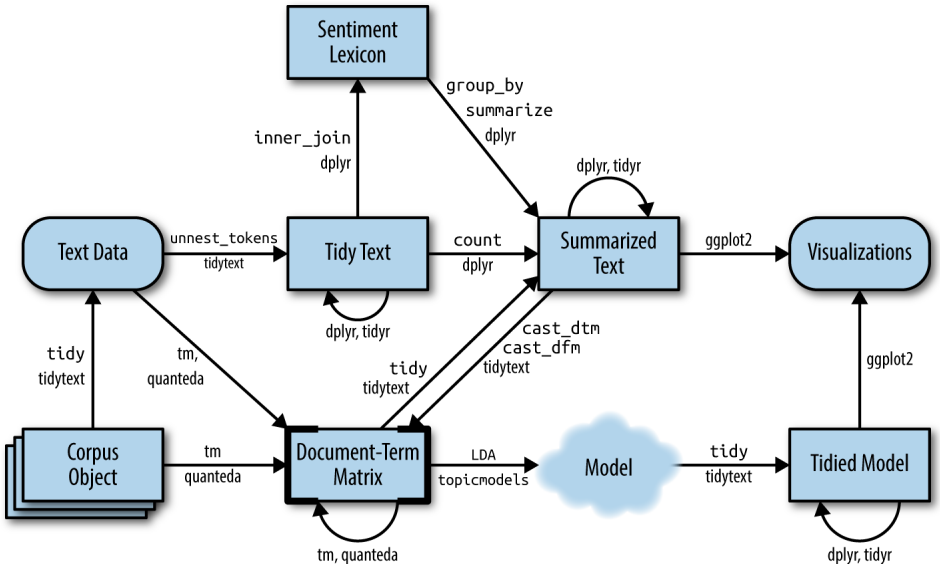


Figure 7.1: Silge & Robinson 2022: A flowchart of a text analysis that incorporates topic modeling. The topicmodels package takes a Document-Term Matrix as input and produces a model that can be tided by tidytext, such that it can be manipulated and visualized with dplyr and ggplot2.

```
#Create Elegant Data Visualisations Using the Grammar of Graphics
library(ggplot2)
library(ggthemes)
# Reddit Data Extraction Toolkit
library(RedditExtractoR)
# Flexibly Reshape Data
library(reshape2)
# Estimation of Structural Topic Models
library (stm)
```

7.2 Data

Reddit is a social news website where you can find posts about almost anything. Reddit has a huge user base and is increasingly used One of the most interesting aspects of Reddit is the comments that accompany posts. Redditors are known for their brutal honesty and often provide interesting opinions that you wouldn't otherwise find. Reddit also has a **plug-n-play** R package which makes is very easy to `get` Reddit data via API.

The key is to find URLs to Reddit threads of interest. There are 2 available search strategies:

by keywords and by home page. Using a set of *keywords* can help you narrow down your search to a topic of interest that crosses multiple subreddits whereas searching by *home page* can help you find, for example, top posts within a specific subreddit.

If you want to source your own Reddit data, comment out the code below and change keywords.

```
#This function takes a collection of URLs and returns a list with 2 data frames: 1. a data
#urls <- find_thread_urls(keywords = "fertility", sort_by = "top", subreddit = NA, period

#This function GETs the data.
#fertility_data <- get_thread_content(urls$url)

# The below code simply creates data frames for the threads and the comments and saves the

#threads <- pandemic_babies_data$threads
#comments <- pandemic_babies_data$comments
#write.csv(threads, "data/topic-modelling/threads.csv", row.names = FALSE)
#write.csv(comments, "data/topic-modelling/comments.csv", row.names = FALSE)
```

First we import the data we will be working with. You can either use the same data as used below or find other data sourced from reddit here.

```
comments <- read.csv("data/topic-modelling/comments.csv", header = TRUE)

head(comments)
```

```
                                                                                    url
1 https://www.reddit.com/r/PrequelMemes/comments/zjcuzy/begun_the_clone_war_has/
2 https://www.reddit.com/r/PrequelMemes/comments/zjcuzy/begun_the_clone_war_has/
3 https://www.reddit.com/r/PrequelMemes/comments/zjcuzy/begun_the_clone_war_has/
4 https://www.reddit.com/r/PrequelMemes/comments/zjcuzy/begun_the_clone_war_has/
5 https://www.reddit.com/r/PrequelMemes/comments/zjcuzy/begun_the_clone_war_has/
6 https://www.reddit.com/r/PrequelMemes/comments/zjcuzy/begun_the_clone_war_has/
             author       date  timestamp score upvotes downvotes golds
1    LaLiLuLeLo9001 2022-12-11 1670802198   223     223         0     0
2 Obiwan-Kenobi-Bot 2022-12-11 1670802222   113     113         0     0
3    LaLiLuLeLo9001 2022-12-11 1670802288    75      75         0     0
4 Obiwan-Kenobi-Bot 2022-12-11 1670802305    63      63         0     0
5         Chung_Soy 2022-12-12 1670821362    37      37         0     0
6 Obiwan-Kenobi-Bot 2022-12-12 1670821387    35      35         0     0

1
```

```
2                                                                                That is kind of y
3
4  We all have difficult times in our lives. It is not easy to maintain a positive self-imag
5
6                                                          Difficult times are part of life, l
   comment_id
1           1
2         1_1
3       1_1_1
4     1_1_1_1
5   1_1_1_1_1
6 1_1_1_1_1_1
```

We now have a data frame with authors, dates and comments.

7.2.1 Text data structures

However, we need to create a Corpus style object to preserve both both the full text of our Reddit comments and the metadata to eventually move to a Document-Term matrix used Topic Modelling. We are going to be using the package `tidytext`.

```
tidy_fertility_reddit <- comments %>% # Takes comments dataframe
  select(timestamp, comment) %>% # Breaks out the timestamp (like a unique idenified) and
  unnest_tokens("word", comment) # Passes the "word" token and the name of the variable wh

head(tidy_fertility_reddit) # Checks out the first 5 words and the dataframe format
```

```
   timestamp     word
1 1670802198      i'm
2 1670802198    sorry
3 1670802198      are
4 1670802198      you
5 1670802198 implying
6 1670802198      obi
```

The `tidytext` format is very useful because once the text has been tidy-ed, regular R functions can be used to analyze it instead of the specialized functions. For example, to count the most popular words in our Reddit, we can can un-comment the following:

```
#tidy_fertility_reddit %>%
#  count(word) %>%
#  arrange(desc(n))
```

7.2.2 Basic text data principles

Before we can run any type of analysis, we first need to decide precisely which type of text should be included in our analyses. For example, as the code above showed, common words such as "the", "and" and "that" are most likely not very informative. Usually, words such as "the" will not be informative for our quantitative text analysis, but how many times reddit comments use the word "abortion" might be very relevant to an analysis about pro-choice discourses.

`Stopwords`

```
data("stop_words") # Stopwords in tidytext package
tidy_fertility_reddit_clean <- tidy_fertility_reddit %>%
  anti_join(stop_words) #using anti-join to remove words
```

```
Joining with `by = join_by(word)`
```

`Punctuation and numbers`

An advantage of `tidytext` is that it removes punctuation automatically. There is also very easy in `tidytext` to remove all numeric digits. We can use basic grep commands (note the "\b\d+\b" text here tells R to remove all numeric digits and the '-' sign means grep excludes them rather than includes them). Grep (Global Regular Expression print) commands are used in searching and matching text files contained in the regular expressions.

```
tidy_fertility_reddit_clean<-tidy_fertility_reddit_clean[-grep("\\b\\d+\\b", tidy_fertilit

# Eliminate some specific words
tidy_fertility_reddit_clean <- tidy_fertility_reddit_clean %>%
  filter(!(word %in% c("https", "www.reddit.com", "comments", "gt", "don", "roe", "post",

# Replace some words with others (manual cleaning)
tidy_fertility_reddit_clean <- tidy_fertility_reddit_clean %>%
  mutate(word = if_else(word == "children", "child", word)) %>%
  mutate(word = if_else(word == "kids", "child", word)) %>%
  mutate(word = if_else(word == "pregnancy", "pregnant", word))
```

We could always do more cleaning.

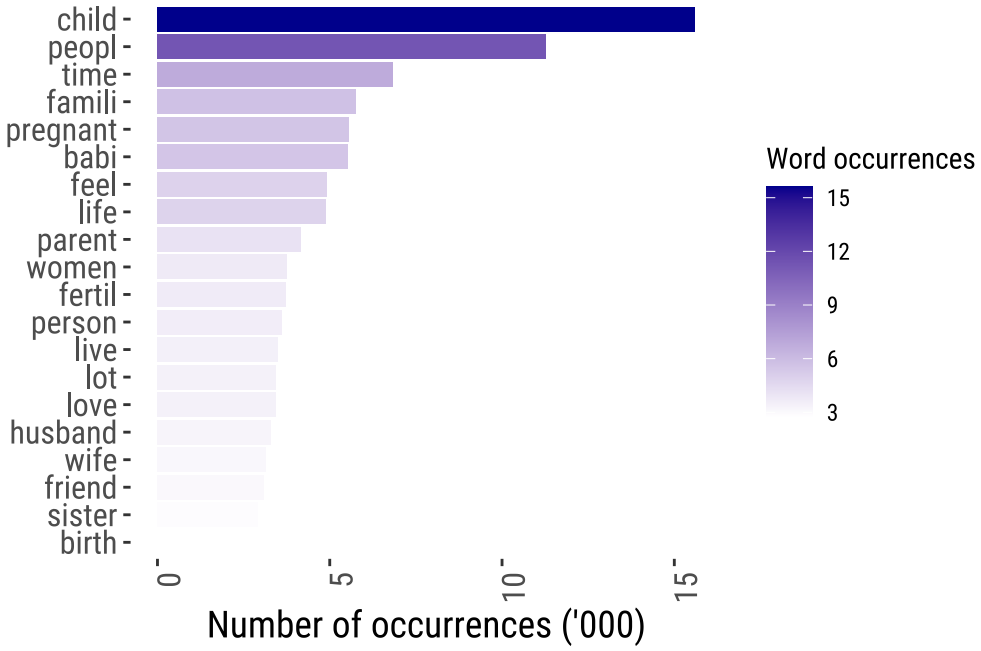
`Stemming`

Stemming reduces words to most basic forms. A final common step in text-pre processing is stemming. Stemming a word refers to replacing it with its most basic conjugate form. For example the stem of the word "typing" is "type." Stemming is common practice because we don't want the words "type" and "typing" to convey different meanings to algorithms that we will soon use to extract latent themes from unstructured texts. `Tidytext` includes the `wordStem` function:

```
  tidy_fertility_reddit_clean<-tidy_fertility_reddit_clean %>%
      mutate_at("word", ~wordStem(., language = "en"))
```

Analysing word frequencies is often the first stop in text analysis. We can easily do this using ggplot. Like in sentiment analysis, let's visualise the 20 most common words used on reddit regarding fertility-related topics.

```
tidy_fertility_reddit_clean %>%
  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()
```



The Document-Term Matrix (DTM)

Finally, we transform our data into a document-term matrix which is the format we will be needing for quantitative text analysis. This is a matrix where each word is a row and each column is a document. The number within each cell describes the number of times the word appears in the document. Many of the most popular forms of text analysis, such as topic models, require a document-term matrix.

To create a DTM in `tidytext` we can use the following code:

```
tidy_fertility_DTM<-
  tidy_fertility_reddit_clean %>%
  count(timestamp, word) %>%
  cast_dtm(timestamp, word, n)

inspect(tidy_fertility_DTM[1:5,3:8])
```

```
<<DocumentTermMatrix (documents: 5, terms: 6)>>
Non-/sparse entries: 6/24
Sparsity           : 80%
Maximal term length: 7
Weighting          : term frequency (tf)
Sample             :
```

```
           Terms
Docs        act aren awkward dis normal wish
  1645984812   0    0       0   0      0    0
  1645984815   0    0       0   1      0    0
  1645984839   1    1       0   0      1    1
  1645984869   0    0       1   0      0    0
  1645984938   0    0       0   0      0    0
```

7.3 Topic Modelling

After pre-processing out text, we can focus on the key of this chapter: discussions around fertility and the pandemic's influence on fertility rates. We do this by using topic modelling.

To start, we will use the DTM we created from the reddit data and the `LDA()` function from the `topicmodels` package, setting k = 5, to create a five-topic LDA model. Almost any topic model in practice will use a larger k, but we will soon see that this analysis approach extends to a larger number of topics.

This function returns an object containing the full details of the model fit, such as how words are associated with topics and how topics are associated with documents.

```
# set a seed so that the output of the model is predictable

Reddit_topic_model <- LDA(tidy_fertility_DTM,
              k = 5, # number of presumed topics
              control = list(seed = 541)) # important if you want this to be reproducible,

Reddit_topic_model
```

```
A LDA_VEM topic model with 5 topics.
```

Fitting the model was the "easy part": the rest of the analysis will involve exploring and interpreting the model using `tidy` functions from the `tidytext` package.

7.3.1 Word-topic probabilities

We can use the `tidy()` function, originally from the broom package Robinson 2017, for tidying model objects. The `tidytext` package provides this method for extracting the per-topic-per-word probabilities, called ("beta"), from the model.

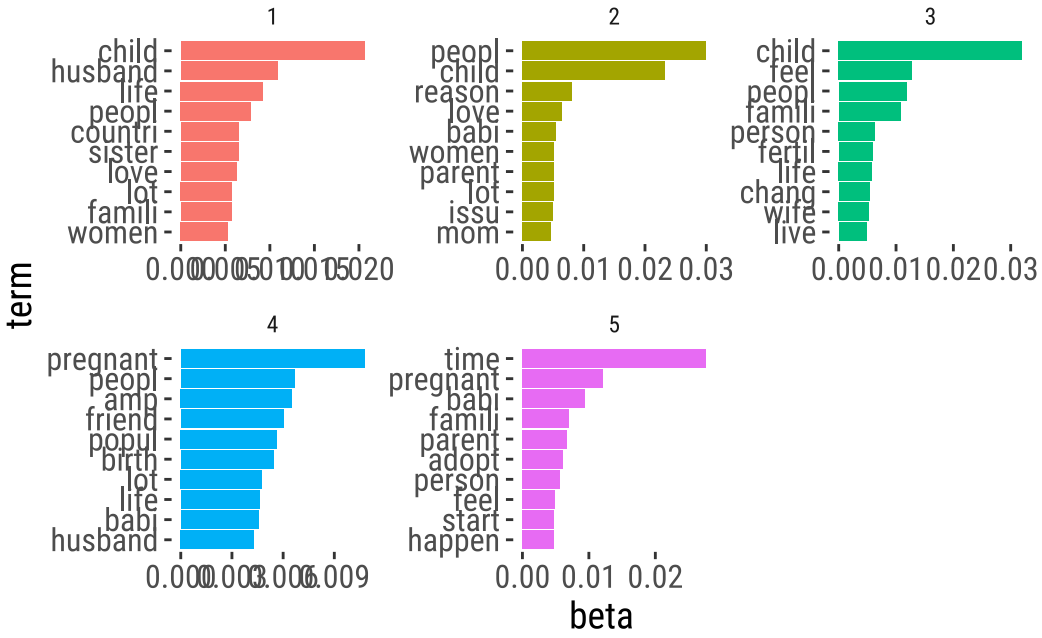
```r
ap_topics <- tidy(Reddit_topic_model, matrix = "beta")
ap_topics
```

```
# A tibble: 162,320 x 3
   topic term       beta
   <int> <chr>     <dbl>
 1     1 carri 0.000567
 2     2 carri 0.00115
 3     3 carri 0.0000683
 4     4 carri 0.000402
 5     5 carri 0.00117
 6     1 deal  0.000830
 7     2 deal  0.00164
 8     3 deal  0.00203
 9     4 deal  0.000326
10     5 deal  0.000681
# i 162,310 more rows
```

This has turned the model into a one-topic-per-term-per-row format. For each combination, the model computes the probability of that term being generated from that topic. CHANGE: For example, the term "aaron" has a 1.686917 × 10−12 probability of being generated from topic 1, but a 3.8959408 × 10−5 probability of being generated from topic 2.

Then there are different options. We could use dplyr's slice_max() to find the 10 terms that are most common within each topic. As a tidy data frame, this lends itself well to a ggplot2 visualization.

```r
ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

ap_top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered() +
  theme_tufte2()
```



Have we defined too many topics? Do we need to increase the number of words per topic. We can see that the Topic 1 focuses on "pregancy" and "adoption", while Topic 3 is probably addressing "legal" questions around fertility. We would need to clean the data further to identify better patterns.

7.3.2 Greatest differences in β

We can consider the terms that had the greatest difference in between Topic 1 and Topic 3. This can be estimated based on the log ratio of the two: $\log_2(\frac{\beta_2}{\beta_1})$ (a log ratio is useful because it makes the difference symmetrical: β_2 being twice as large leads to a log ratio of 1, while β_1 being twice as large results in -1). To make sure we pick up relevant words, we can filter for relatively common words, such as those that have a β greater than 1/1000 in at least one topic.

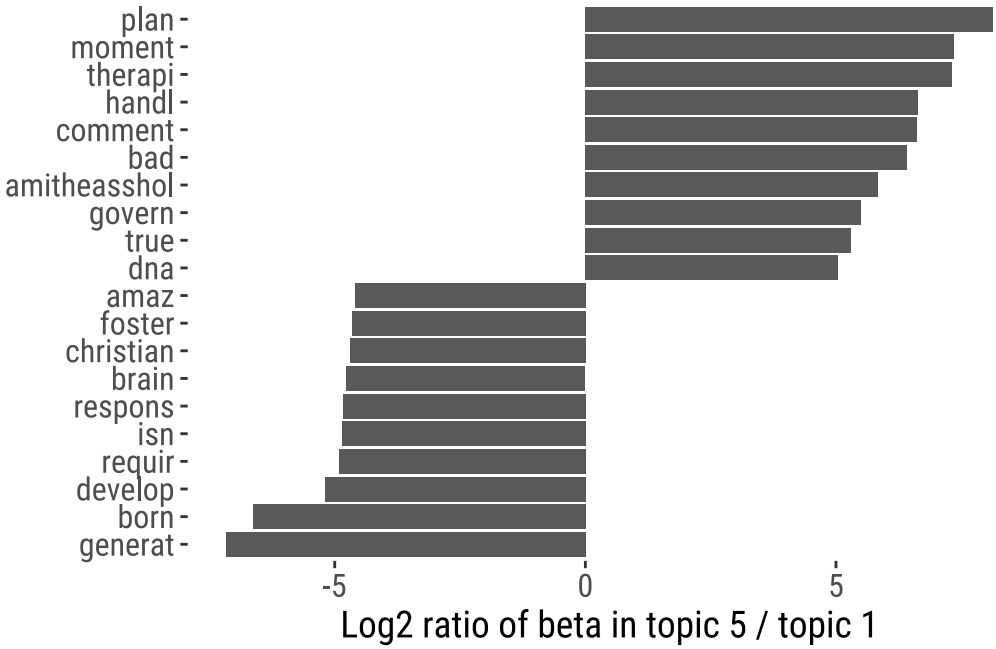
```
ap_topics <- ap_topics %>%
  filter(topic == 1 | topic == 5) # Keeping just the topics of interest

beta_wide <- ap_topics %>%
  mutate(topic = paste0("topic", topic)) %>%
  pivot_wider(names_from = topic, values_from = beta) %>%
  filter(topic1 > .001 | topic5 > .001) %>% # Beta greater than 1/1000 in at least one top
```

```
    mutate(log_ratio = log2(topic5 / topic1)) # Calculate Log ratio

  beta_wide
```

```
# A tibble: 275 x 4
   term     topic1    topic5 log_ratio
   <chr>     <dbl>     <dbl>     <dbl>
 1 carri  0.000567 0.00117       1.05
 2 lot    0.00574  0.000850     -2.76
 3 run    0.00134  0.0000847    -3.98
 4 child  0.0206   0.00159      -3.70
 5 enjoy  0.000929 0.00102       0.131
 6 fertil 0.00430  0.00362      -0.249
 7 grow   0.00193  0.00236       0.289
 8 hous   0.00166  0.000110     -3.92
 9 life   0.00923  0.00209      -2.14
10 mean   0.000899 0.00118       0.396
# i 265 more rows
```

The words with the greatest differences between the Topic 1 and Topic 3:

```
  beta_wide %>%
    group_by(direction = log_ratio > 0) %>%
    slice_max(abs(log_ratio), n = 10) %>%
    ungroup() %>%
    mutate(term = reorder(term, log_ratio)) %>%
    ggplot(aes(log_ratio, term)) +
    geom_col() +
    labs(x = "Log2 ratio of beta in topic 5 / topic 1", y = NULL) +
    theme_tufte2()
```



We can see that the words more common in topic 3 include words such as "sperm", "embryo" and "response" suggesting we may be picking up medical discussion around fertility. Whereas Topic 1 is more centred around "pregnancy", "parents" and "divorce" suggesting socio-economic …. More exploration would be warranted here…

7.3.3 Structural Topic Modelling

The `stm` package has some text pre-processing functions integrated in it. Similar to the steps we did manually in the previous section. The `textProcessor` function automatically removes a) punctuation; b) stop words; c) numbers, and d) stems each word. The function requires us to specify the part of the dataframe where the documents we want to analyze are documents), and requires us to name the dataset where the rest of the meta data live (pandemic_threads). Notice what happens in your console while function `textProcessor`.

```
pandemic_threads <- read.csv("data/topic-modelling/pandemicthreads.csv", header = TRUE)

head(pandemic_threads)
```

```
1                              https://www.reddit.com/r/entitledparents/comments/v2kalc/am_i
2        https://www.reddit.com/r/BBBY/comments/1144ioj/today_21623_sue_gove_said_bbby_has_a
```

```
3                    https://www.reddit.com/r/JUSTNOMIL/comments/uizf4z/the_other_son_is_the_
4 https://www.reddit.com/r/hiphopheads/comments/10c2n82/album_of_the_year_25_kendrick_lamar_m
5                 https://www.reddit.com/r/collapse/comments/y4mqrd/last_week_in_collapse_octo
6     https://www.reddit.com/r/bridezillas/comments/yyda0z/slavedriver_bridezilla_starved_my_
                author       date  timestamp
1     user_not_found01 2022-06-01 1654099230
2      DroppingVittles 2023-02-16 1676590184
3 Fair_Personality_122 2022-05-05 1651762956
4   freshsupreme_acist 2023-01-14 1673735518
5   LastWeekInCollapse 2022-10-15 1665837036
6           Azazeru921 2022-11-18 1668753411
                                                                                  title
1                                                                    Am I overreacting?
2  Today (2/16/23) Sue Gove said BBBY has a new "supplier promise" and more...
3                                                 The other son is the golden child
4    Album of the Year #25 : Kendrick Lamar - Mr Morale &amp;amp; The Big Steppers
5                                      Last Week in Collapse: October 8-14, 2022
6 Slavedriver Bridezilla, starved my mother while she was staying at her place

                                                                                  text
1
1 store format 2n the 1970s and \03180s\024back then it was just \034Bed &amp;amp; Bath\035\024as 1.0, and its 1
3
4 Artist : Kendrick Lamar\n\nAlbum : \034Mr. Morale &amp;amp; The Big Steppers\n\nApple download
news/world-europe-63217467) nicknamed \034General Armageddon.\035 [Power was restored](https://www.reuters.c
ants but my sister told me that the materials she wants are very expensive. My sis gave her an estimate and
        subreddit score upvotes downvotes up_ratio total_awards_received golds
1 entitledparents   528     528         0     0.99                     1     0
2            BBBY   527     527         0     0.97                     0     0
3       JUSTNOMIL   520     520         0     0.97                     0     0
4     hiphopheads   515     515         0     0.84                     0     0
5        collapse   515     515         0     0.98                    10     0
6     bridezillas   514     514         0     0.96                     0     0
  cross_posts comments
1           0       42
2           0       48
3           0       53
4           1      293
5           0       36
6           0       48
```

```
processed <- textProcessor(pandemic_threads$text, metadata = pandemic_threads)
```

```
Building corpus...
Converting to Lower Case...
Removing punctuation...
Removing stopwords...
Removing numbers...
Stemming...
Creating Output...
```

The `stm` package also requires us to store the documents, meta data, and "vocab" in separate objects, essentially a list of words described in the documents.

```
# Eliminates both extremely common terms and extremely rare terms, since such terms make w
out <- prepDocuments(processed$documents, processed$vocab, processed$meta)
```

```
Removing 6469 of 11816 terms (6469 of 71023 tokens) due to frequency
Your corpus now has 218 documents, 5347 terms and 64554 tokens.
```

```
docs <- out$documents
vocab <- out$vocab
meta <-out$meta
```

Then have to make another decision about the number of topics we might expect to find in the corpus. Let's start out with 10. We also need to specify how we want to use the meta data. This model uses the number of "comments". It's important to recognize that the variables selected in this stage can have significant ramifications. If we make the wrong choice, we could potentially miss identifying certain topics that are discussed on both liberal and conservative blogs or mistakenly categorize them as distinct subjects.

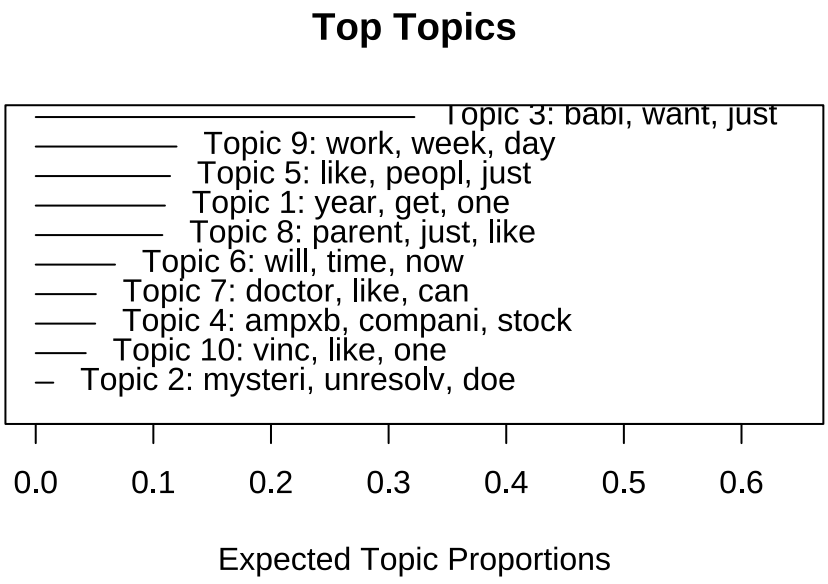
In addition, the `stm` package offers an argument that permits the specification of the desired type of initialization or randomization. For our purposes, we have chosen to use spectral initialization. Please see Bail, C. 2020 for more.

This below code may take some time if you are running it on a large body. You can read more about each function in the package documentation.

```
First_STM <- stm(documents = out$documents, vocab = out$vocab,
              K = 10,
              prevalence =~ comments,
              max.em.its = 75, data = out$meta,
              init.type = "Spectral", verbose = FALSE)
```

We start by inspecting our results by browsing the top words associated with each topic. The `stm` package has a useful function that visualizes these results called plot:

```
plot(First_STM)
```



The visualization provides information on both the occurrence rate of the topic across the entire corpus and the top three words that are linked to the topic. As you will notice, in a second iteration of the model we may want to exclude words such as "like".

Some topics seem plausible, but many that do not seem very coherent or meaningful. You may want to improve your topic classification with more than one variable in the `prevalence` comments. Please see Bail, C. 2020 for more.

7.3.4 Limitations of Topic Models

For various reasons, topic models have become a conventional tool for quantitative text analysis. Depending on the application, they can be more advantageous than simplistic word frequency or dictionary-based methods. Generally, topic models yield optimal outcomes when utilized on texts that are moderately lengthy and have a regular format.

On the other hand, topic models have a number of important limitations. To start, the term "topic" is somewhat vague, and it is now evident that topic models cannot generate

extremely refined classifications of texts. Furthermore, if topic models are incorrectly perceived as an unbiased depiction of a text's meaning, they can be easily misused. Once more, these instruments might be more accurately depicted as "tools for reading." It is not advisable to excessively interpret the outcomes of topic models unless the researcher has solid theoretical prior knowledge regarding the number of topics in a particular corpus, or if the researcher has thoroughly verified the results of a topic model using both quantitative and qualitative methodologies described earlier.

7.4 Questions

For the second assignment, we will focus on the United Kingdom as our geographical area of analysis. As for Sentiment Analysis chapter, we will use a dataset of tweets about migration posted by users in the United Kingdom during February 24th 2021 to July 1st 2022.

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

1. Prepare the Twitter data so that it can be analyzed in the `topicmodels` package
2. Run three models and try to identify an appropriate value for k (the number of topics).
3. Create a chart of greatest differences between two relevant topics you have identified.
4. Use the `full_place_name` or `lat` and `long` variables as meta data to create classification between different types of places in the UK. For example: urban/rural, classified by population/density, or simply between different regions. There are plenty of ways you can divide the tweets geographically, see some easy examples here. Then explore whether there are differences in topics according to different locations in the UK using the `stm` package.
5. BONUS QUESTION: Discuss the limitations of topic models on short texts, such as Tweets. There have been a number of recent attempts to address this problem, and Graham Tierney has developed a very nice solution called stLDA-C.

Analyse and discuss: a) whether there are different topics related to migration that emerge, and what these are. b) how migration topics vary spatially.

The below code helps you geo-localise the Twitter data. You could then perform a spatial join to another sf objects that has population density or classifies areas in the UK.

```
library(sf)

# subset the data frame to remove rows with missing values in x and y columns
twitter_df_clean <- twitter_df[complete.cases(twitter_df[, c("lat", "long")]), ]
```

```r
twitter_df_clean <- twitter_df_clean %>%
  sf::st_as_sf(coords = c(4,5))  %>%  # create pts from coordinates
  st_set_crs(4326)  # set the original CRS

plot(twitter_df_clean$geometry)
```

```r
# Example spatial join. You can also specify other join types such as st_contains, st_with
# cities <- st_join(point, cities, join = st_intersects)
```

8 Modelling Time

Time is a critical variable in many areas of research, and its modeling can help us gain insights into underlying phenomena. Timeseries data has proven to be a valuable tool in understanding the COVID-19 pandemic. By analyzing data over time, researchers have been able to track changes in the spread of the virus, the effectiveness of interventions such as social distancing and vaccination campaigns, and the impact of the pandemic on various social and economic indicators. With the increasing availability of digital footprint data, which includes information both of offline behaviour such as visits to grocery shops and parks, as well as online behavior such as internet searches and social media activity, researchers have had new opportunities to understand the pandemic's impact on individuals and communities (Ugolini et al. 2020; Schleicher 2020) and (Cinelli et al. 2020). These data sources provide insight into people's attitudes, concerns, and behaviors during the pandemic, as well as their response to interventions and policies. Together, timeseries and digital footprint data offer powerful tools for understanding the complex dynamics of the COVID-19 pandemic and developing effective responses.

This chapter will introduce modelling time descriptively using linear trends, quadratic trends, and splines. It will also provide and introduction to modelling data over time and space.

8.1 Dependencies

```
# Data
library(sf)
library(readr)
library(tidyverse)

# Dates and Times
library(lubridate)

# Regression Spline Functions and Classes
library(splines)

# graphs
library(ggplot2)
library(dygraphs)
```

```
library(plotly)
library(ggthemes)
library(ggpmisc)
library(gridExtra)
library(viridis)
library(ggformula)
library(ggimage)
library(grid)
```

8.2 Data

We will use a sample of Google Mobility data. Google has made available Community Mobility data to provide insights into what changed in response to policies aimed at combating COVID-19. The data also has accompanying reports which analyse movement trends over time by geography, across different categories of places such as retail and recreation, groceries and pharmacies, parks, transit stations, workplaces, and residential. Data is available from 2020 and 2022. Community Mobility data is no longer being updated as of 2022-10-15. All historical data will remain publicly available.

Location accuracy and the understanding of categorised places varies from region to region, so it is not recommend the data is used to compare changes between countries, or between regions with different characteristics (e.g. rural versus urban areas). Region that do not have statistically significant levels of data have been left out of the report.

Some important facts about the data:

- Changes for each day are compared to a baseline value for that day of the week. The baseline is the median value.
- The data that is included in the calculation depends on user settings, connectivity and whether it meets our privacy threshold.
- If the privacy threshold isn't met (when somewhere isn't busy enough to ensure anonymity) a change for the day is not shown.

For more about this data please see here.

8.3 Modelling Time

First we import the data we will be working with. We will be looking at Google Mobility data for Italy.

```
mobility_it <- read.csv("data/longitudinal-1/2022_IT_Region_Mobility_Report.csv", header =
```

Check out variables in the dataframe

```
# Check out variables in the dataframe
colnames(mobility_it)
```

```
 [1] "country_region_code"
 [2] "country_region"
 [3] "sub_region_1"
 [4] "sub_region_2"
 [5] "metro_area"
 [6] "iso_3166_2_code"
 [7] "census_fips_code"
 [8] "place_id"
 [9] "date"
[10] "retail_and_recreation_percent_change_from_baseline"
[11] "grocery_and_pharmacy_percent_change_from_baseline"
[12] "parks_percent_change_from_baseline"
[13] "transit_stations_percent_change_from_baseline"
[14] "workplaces_percent_change_from_baseline"
[15] "residential_percent_change_from_baseline"
```

Long data vs. Wide data

Long data and wide data are two different formats used to store and organize data in a tabular form. Wide data is a format where each variable is stored in a separate column, and each observation or time point is stored in a separate row. This format is often useful when the number of variables is small compared to the number of observations, and is often used for descriptive statistics and exploratory data analysis.

```
        dates           A          B
1  2022-01-01 -0.56047565  1.2240818
2  2022-01-02 -0.23017749  0.3598138
3  2022-01-03  1.55870831  0.4007715
4  2022-01-04  0.07050839  0.1106827
5  2022-01-05  0.12928774 -0.5558411
6  2022-01-06  1.71506499  1.7869131
7  2022-01-07  0.46091621  0.4978505
8  2022-01-08 -1.26506123 -1.9666172
9  2022-01-09 -0.68685285  0.7013559
10 2022-01-10 -0.44566197 -0.4727914
```

Long data, on the other hand, is a format where each variable is represented by two or more columns: one column for the variable name and another for the variable values. This format is often useful when we have many variables or when we want to perform statistical analysis.

```
        dates series       value
1  2022-01-01      A -0.56047565
2  2022-01-01      B  1.22408180
3  2022-01-02      A -0.23017749
4  2022-01-02      B  0.35981383
5  2022-01-03      A  1.55870831
6  2022-01-03      B  0.40077145
7  2022-01-04      A  0.07050839
8  2022-01-04      B  0.11068272
9  2022-01-05      A  0.12928774
10 2022-01-05      B -0.55584113
11 2022-01-06      A  1.71506499
12 2022-01-06      B  1.78691314
13 2022-01-07      A  0.46091621
14 2022-01-07      B  0.49785048
15 2022-01-08      A -1.26506123
16 2022-01-08      B -1.96661716
17 2022-01-09      A -0.68685285
18 2022-01-09      B  0.70135590
19 2022-01-10      A -0.44566197
20 2022-01-10      B -0.47279141
```

Both wide and long data formats have their own advantages and disadvantages, and the choice between them often depends on the specific data and the analysis that is planned. Transforming data from one format to another is a common task in data processing and analysis, and can be accomplished using various data manipulation tools in R, such as `tidyr` and `reshape2` packages. Look back at the Italian google mobility data to check which format it is in.

Date format

Time series aim to study the evolution of one or several variables through time. Several packages that are part of the `tidyverse` family will help you analyse time series data in `R`. The `lubridate`package is your best friend to deal with the date format. `ggplot2` will allow you to plot it efficiently. `dygraphs` will also help build attractive interactive charts.

Building time series requires the time variable to be at the date format. The first step of your analysis must be to double check that R read your data correctly, i.e. at the date format. This is possible thanks to the str() function:

```
# Checking date format
str(mobility_it)
```

```
'data.frame':   36576 obs. of  15 variables:
 $ country_region_code                                : chr  "IT" "IT" "IT" "IT" ...
 $ country_region                                     : chr  "Italy" "Italy" "Italy" "Italy"
 $ sub_region_1                                       : chr  "ALL" "ALL" "ALL" "ALL" ...
 $ sub_region_2                                       : chr  "" "" "" "" ...
 $ metro_area                                         : logi  NA NA NA NA NA NA ...
 $ iso_3166_2_code                                    : chr  "" "" "" "" ...
 $ census_fips_code                                   : logi  NA NA NA NA NA NA ...
 $ place_id                                           : chr  "ChIJA9KNRIL-1BIRb15jJFz1LOI" "Ch
 $ date                                               : chr  "01/01/2022" "02/01/2022" "03/01/
 $ retail_and_recreation_percent_change_from_baseline: int  -65 -27 -13 -13 -10 -30 -20 -27 -
 $ grocery_and_pharmacy_percent_change_from_baseline : int  -80 7 28 30 42 -24 25 6 -7 19 ..
 $ parks_percent_change_from_baseline                : int  21 12 19 17 10 42 13 3 -36 -18 .
 $ transit_stations_percent_change_from_baseline     : int  -49 -18 -36 -37 -37 -50 -38 -30 -
 $ workplaces_percent_change_from_baseline           : int  -69 -13 -42 -41 -42 -78 -48 -27 -
 $ residential_percent_change_from_baseline          : int  13 6 13 13 12 23 16 9 10 10 ...
```

This is already looking fine in the google mobility data, but in many other cases the `lubridate` package is such a life saver. It offers several function which name are composed by 3 letters: year (`y`), month (`m`) and day (`d`). Have a look at `lubridate` cheat sheet to see more about converting time variables to useful formats. There is also the anytime() function in the anytime package whose sole goal is to automatically parse strings as dates regardless of the format.

```
#Convert the date column to a date format using the dmy() function
mobility_it$date_fix <- dmy(mobility_it$date)
```

Tidy up some variables.

```
# Rename variables
mobility_it <- mobility_it %>%
  rename(grocery_pharmacy = grocery_and_pharmacy_percent_change_from_baseline,
         parks_percent_change_from_baseline = parks_percent_change_from_baseline,
         transit = transit_stations_percent_change_from_baseline)
```

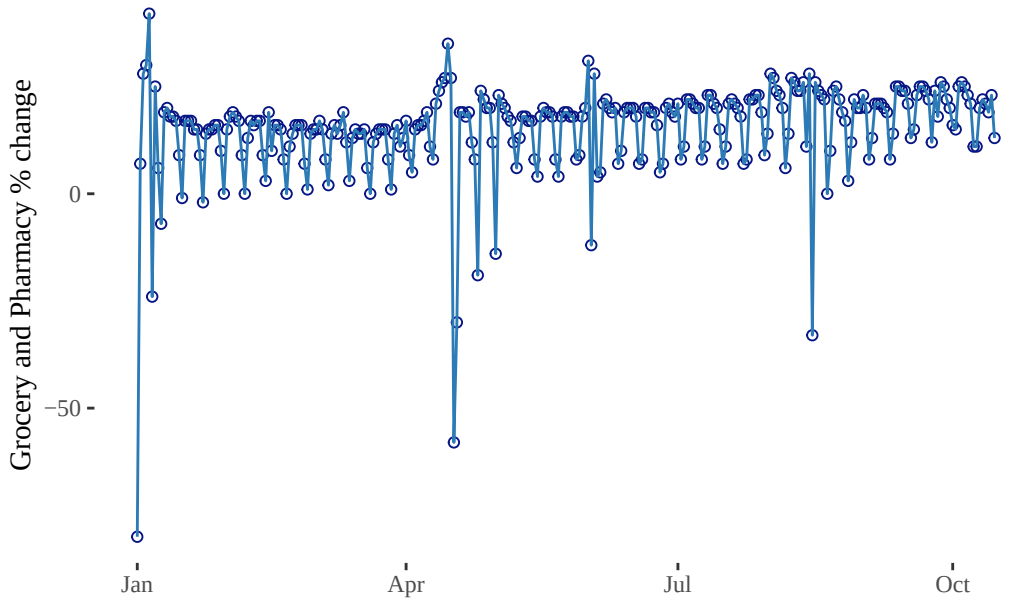
Initial time-series plotting with ggplot2

Let's start easy by keeping only data for the whole of Italy.

```
# Filter the dataframe to keep only the rows for all of Italy
mobility_it_nogeo <-filter(mobility_it, sub_region_1 == "ALL")
```

ggplot2 offers great features when it comes to visualize time series. The date format will be recognized automatically, resulting in a neat x axis labels.

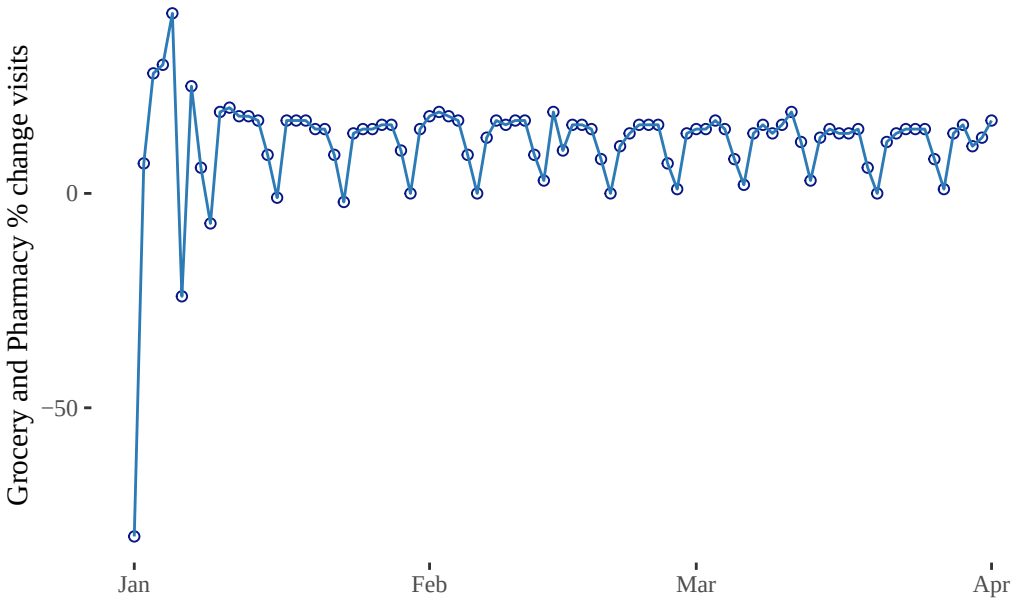
```
# Most basic bubble plot - uncomment to visualise
#timeseries1 <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=grocery_pharmacy)) + ge
#timeseries1

# Adding lines and starting to clean up graph
timeseries2 <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=grocery_pharmacy)) +
  geom_point(color="#061685", shape=1) +
  geom_line(color="#2c7bb6") +
  theme_tufte() +
  xlab("") +
  ylab("Grocery and Pharmacy % change")

timeseries2
```



The ggplot2 package recognizes the date format and automatically uses a specific type of X axis. If the time variable isn't at the date format, this won't work. Always check with

`str(data)` how variables are understood by R. The `scale_x_data()` makes it a breeze to customize those labels.

```
# Use the limit option of the scale_x_date() function to select a time frame in the data:
timeseries3 <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=grocery_pharmacy)) +
  geom_point(color="#061685", shape=1) +
  geom_line( color="#2c7bb6") +
  xlab("") +
  ylab("Grocery and Pharmacy % change visits") +
  theme_tufte() +
  scale_x_date(limit=c(as.Date("2022-01-01"),as.Date("2022-04-01"))) # Limiting between tw

timeseries3
```



`plotly` is also great to turn the resulting chart interactive in one more line of code. With the `ggplotly()` function you can hover circles to get a tooltip, or select an area of interest for zooming. You can zoom by selecting an area of interest. Hover over the line to get exact time and value. Export to a widget with `htmlwidgets`.

```
# Usual chart
timeseries3bis <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=grocery_pharmacy)) +
  geom_point(color="#061685", shape=1) +
```

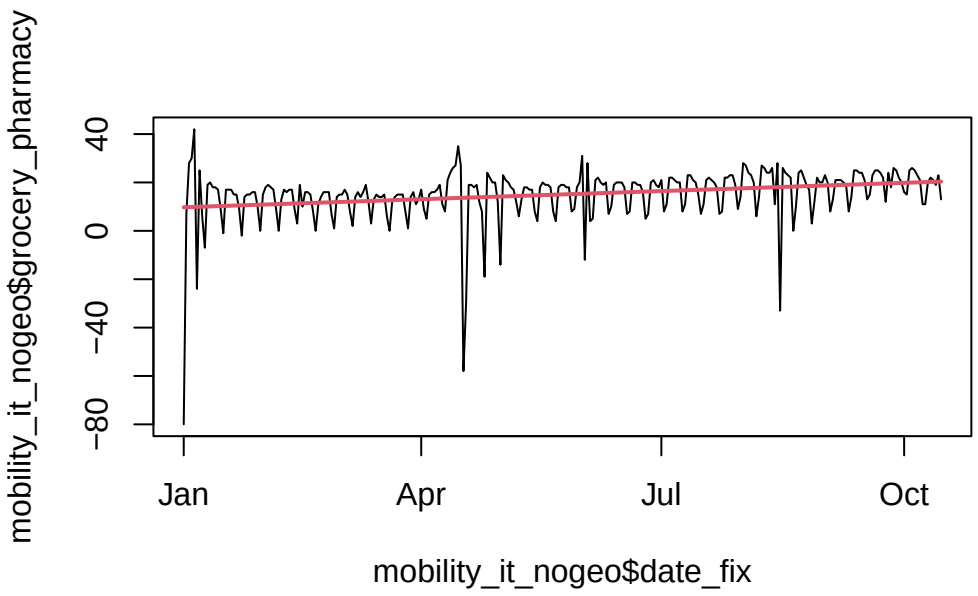
```
  geom_line(color="#2c7bb6") +
  theme_tufte() +
  xlab("") +
  ylab("Grocery and Pharmacy % change")

# Turn it interactive with ggplotly
timeseries_interactive <- ggplotly(timeseries3bis)
timeseries_interactive
```

```
# Of course this needs more cleaning to be a final output

# To save the widget use the library(htmlwidgets)
# saveWidget(p, file=paste0( getwd(), "/HtmlWidget/ggplotlyAreachart.html"))
```

Linear trends

Now let's try to model trends in the data. Let's first consider linear trends. Linear trends are the simplest way to model time as a quantitative variable. We can represent time as a series of evenly spaced points on a graph, and then fit a straight line to those points. The slope of the line tells us the rate at which the variable is changing over time. For example, if we are measuring population counts over time, a positive slope would indicate that a population increase, and a negative slope would indicate a population decrease.

```
# Estimate linear regression model
linear_model <- lm(grocery_pharmacy ~ date_fix, mobility_it_nogeo)

# A simple plot line
plot(mobility_it_nogeo$date_fix,
     mobility_it_nogeo$grocery_pharmacy,
     type = "l")
lines(mobility_it_nogeo$date_fix,
      predict(linear_model),
      col = 2,
      lwd = 2)
```




```
# Extract coefficients of model and print
my_coef <- coef(linear_model)
my_coef
```

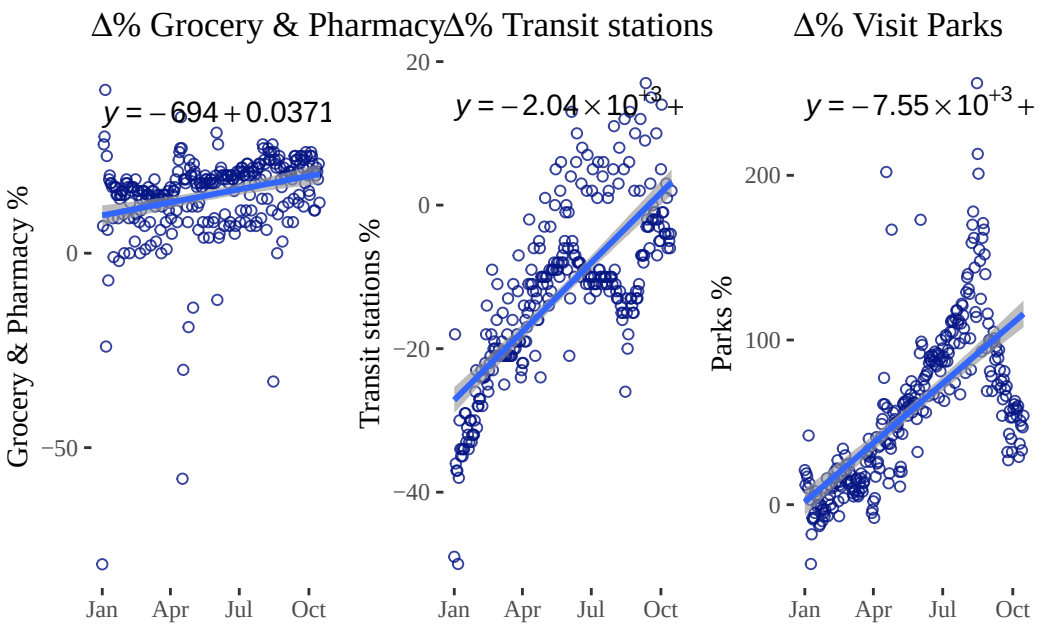
```
  (Intercept)      date_fix
-694.27809786    0.03706687
```

```
# Extract equation of model and print
my_equation <- paste("y =",
                     coef(linear_model)[[1]],
                     "+",
                     coef(linear_model)[[2]],
                     "* x")
my_equation
```

```
[1] "y = -694.278097860568 + 0.037066871224827 * x"
```

Let's plot this with ggplot to fit our line through our points. We can use the package `ggpmisc` to add the equation and R2 with `stat_poly_line()` and `stat_poly_eq`.

```r
# Using geom_smooth() for the linear fit
timeseries4a <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=grocery_pharmacy)) +
  geom_point(color="#061685", shape=1, alpha = 0.8) +
  geom_smooth(formula = y ~ x, method="lm") + # linear regression
  xlab("") +
  ylab("Grocery & Pharmacy %") +
  stat_poly_line() +
  stat_poly_eq(use_label(c("eq", "R2"))) +
 theme_tufte() +
  ggtitle(expression(paste(Delta, "% Grocery & Pharmacy")))

timeseries4b <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=transit)) +
  geom_point(color="#061685", shape=1, alpha = 0.8) +
  geom_smooth(formula = y ~ x, method="lm") + #This is where your linear regression is
  xlab("") +
  ylab("Transit stations %") +
  stat_poly_line() +
  stat_poly_eq(use_label(c("eq", "R2"))) +
 theme_tufte() +
  ggtitle(expression(paste(Delta, "% Transit stations")))

timeseries4c <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=parks_percent_change_fr
  geom_point(color="#061685", shape=1, alpha = 0.8) +
  geom_smooth(formula = y ~ x, method="lm") + #This is where your linear regression is
  xlab("") +
  ylab("Parks %") +
  stat_poly_line() +
  stat_poly_eq(use_label(c("eq", "R2"))) +
 theme_tufte() +
  ggtitle(expression(paste(Delta, "% Visit Parks")))


grid.arrange(timeseries4a, timeseries4b, timeseries4c, ncol=3)
```



Alternatively you can use the `ols()` function aswell followed by the `stat_function()`. See here for more details.

Linear trends have some limitations. In many cases, the relationship between time and the variable we are measuring is not linear, and fitting a straight line may not capture the true nature of the relationship. In such cases, we may need to consider a quadratic trend.

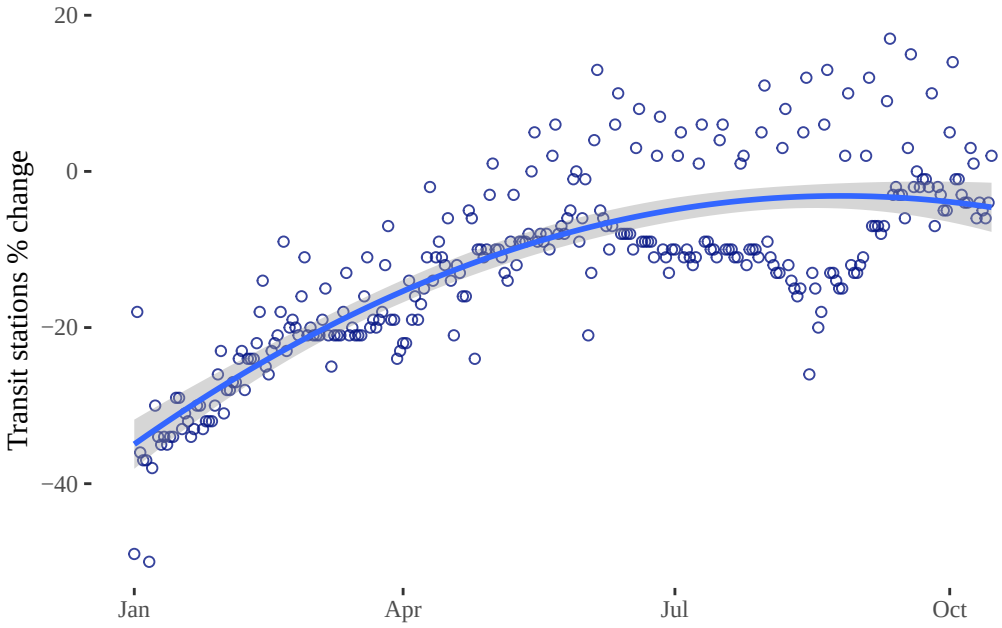
Quadratic trends

Quadratic trends model time as a second-degree polynomial, which allows for a curved relationship between time and the variable we are measuring. Quadratic trends can capture more complex patterns in the data than linear trends, such as a gradual increase followed by a gradual decrease. For example, if we are measuring the number of COVID-19 cases over time, a quadratic trend may capture the initial exponential growth, followed by a flattening out of the curve.

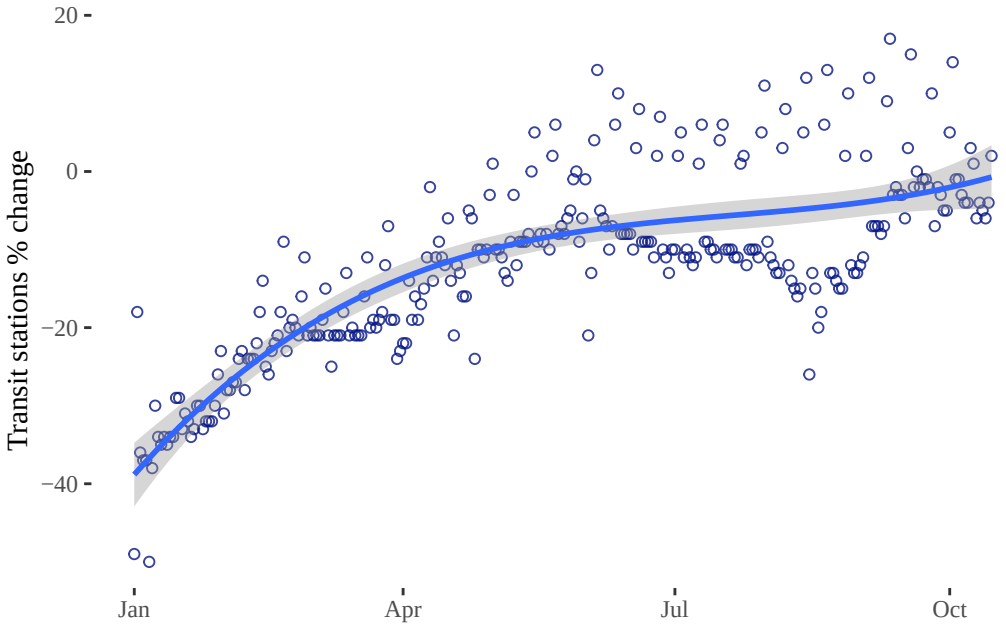
However, quadratic trends also have some limitations. They assume that the relationship between time and the variable we are measuring is symmetrical, which may not always be the case. In addition, they can be difficult to interpret, as the coefficient for the quadratic term does not have a straightforward interpretation.

```
timeseries5 <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=transit)) +
  geom_point(color="#061685", shape=1, alpha = 0.8) +
  geom_smooth(formula = y ~ poly(x,2), method="lm", se = T, level = 0.99) + # 2nd order po
```

```
  xlab("") +
  ylab("Transit stations % change") +
 theme_tufte()

timeseries5
```




```
timeseries6 <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=transit)) +
  geom_point(color="#061685", shape=1, alpha = 0.8) +
  geom_smooth(formula = y ~ poly(x,3), method="lm", se = T, level = 0.99) + # 3rd order po
  xlab("") +
  ylab("Transit stations % change") +
 theme_tufte()

timeseries6
```



Splines

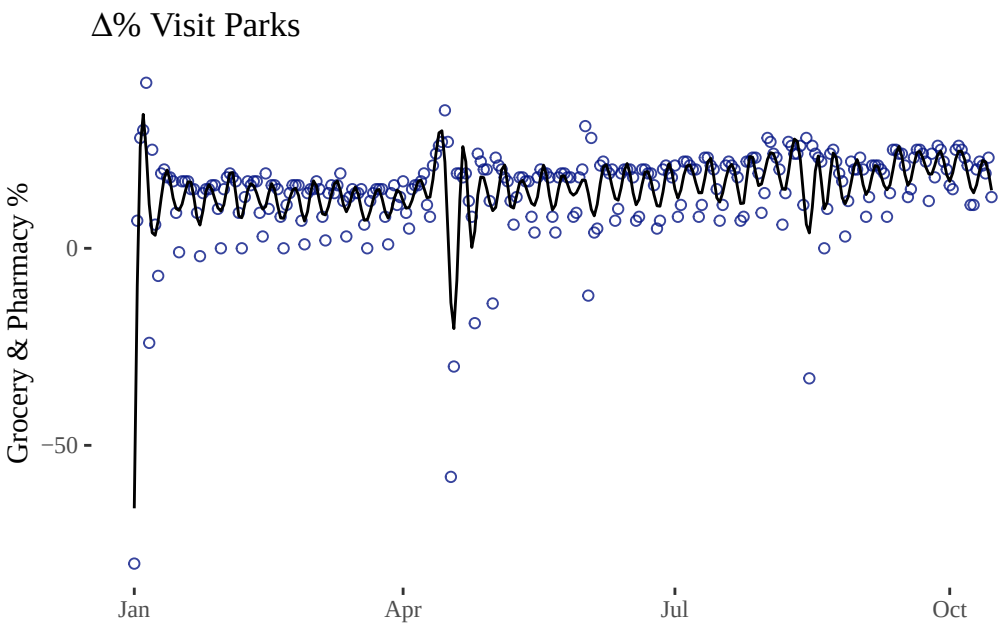
Finally, we have splines. Splines are a more flexible way to model time as a quantitative variable. Splines allow us to fit a piecewise function to the data, where the function is a series of connected polynomial segments. Each segment captures a different part of the relationship between time and the variable we are measuring. Splines can capture more complex patterns in the data than linear or quadratic trends, and they can be customized to fit the specific shape of the relationship we are trying to model.

However, splines also have some limitations. They can be computationally intensive, and the choice of the number and location of the knots (i.e., the points where the polynomial segments connect) can have a big impact on the results. Splines are useful when the underlying function is complex or unknown, and can be used for a variety of applications, including curve fitting, data smoothing, and prediction.

In R, you can plot x vs y using the plot() function. To plot a spline, you can use the spline() function to generate the points for the curve, and then plot the curve using the lines() function.

```
timeseries7 <- ggplot(data = mobility_it_nogeo, aes(x=date_fix, y=grocery_pharmacy)) +
  geom_point(color="#061685", shape=1, alpha = 0.8) +
  ggformula::stat_spline() + # spline
  xlab("") +
```

```
  ylab("Grocery & Pharmacy %") +
 theme_tufte() +
  ggtitle(expression(paste(Delta, "% Visit Parks")))

timeseries7
```



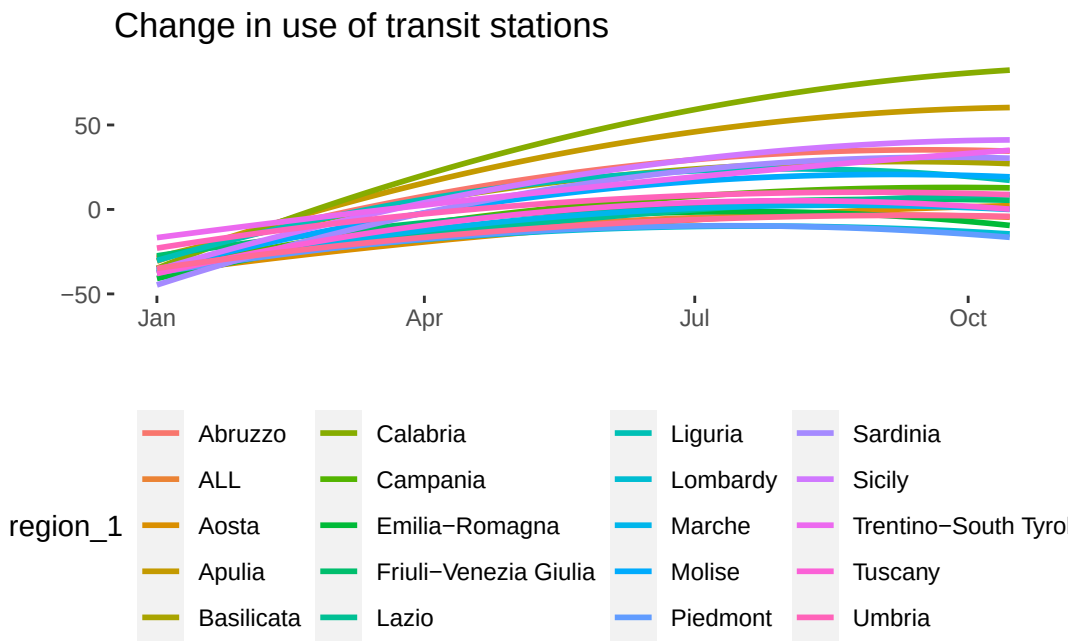
Linear trends, quadratic trends, and splines are all ways to model time as a quantitative variable, each with their own strengths and weaknesses. The choice of method will depend on the specific research question and the shape of the relationship between time and the variable we are measuring. You can find more on descriptive timeseries analysis here.

8.4 Modelling Time and Space

We can also examine the heterogeneity in the data by region. Some regions in Italy were had many more COVID-19 cases than others.

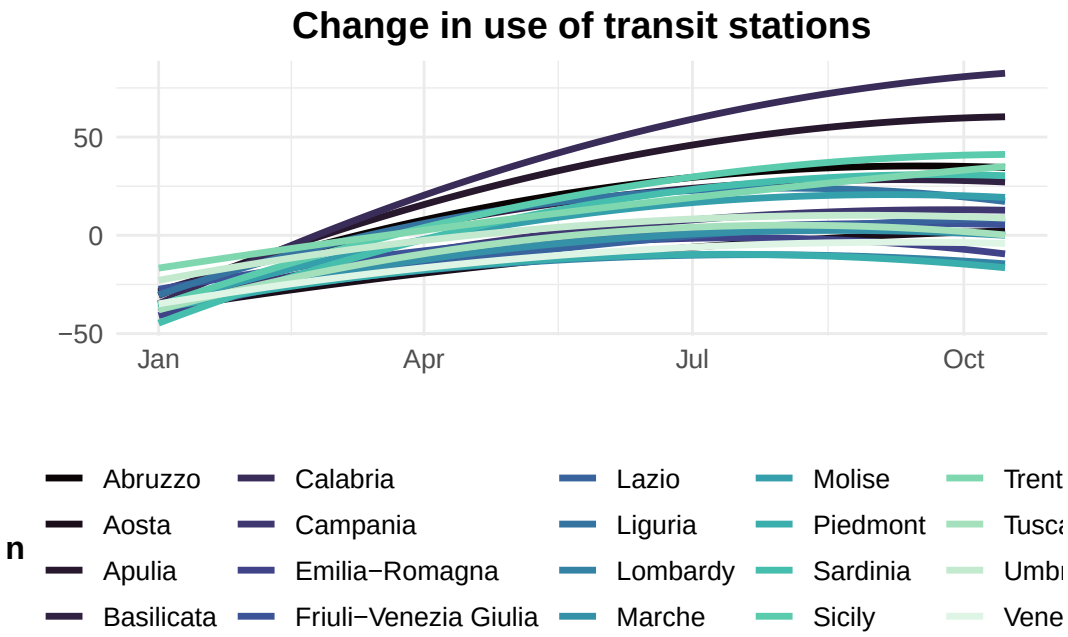
```
# Mobility data by region over time
timeseries_all_1 <- ggplot(data = mobility_it, aes(x=date_fix, y=transit, color = sub_regi
  # geom_point(alpha = 0.8) +
  geom_smooth(method="lm",formula = y ~ poly(x,2), se=F) +
```

```
  theme(
  legend.position = "bottom",
  panel.background = element_rect(fill = NA)) +
  xlab("") +
  ylab("") +
  ggtitle("Change in use of transit stations")

# Initial Graph
timeseries_all_1
```




```
mobility_it_filtered <- mobility_it %>%
  filter(sub_region_1 != "ALL")

# Mobility data by region over time with some edits
timeseries_all_2 <- ggplot(data = mobility_it_filtered, aes(x = date_fix, y = transit, col
  geom_smooth(method = "lm", formula = y ~ poly(x, 2), se = F, size = 1.2) +
  scale_color_viridis(discrete = TRUE, option="mako") +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    plot.title = element_text(size = 14, hjust = 0.5, face = "bold"),
```

```
    axis.text = element_text(size = 10),
    axis.title = element_text(size = 12, face = "bold"),
    legend.text = element_text(size = 10),
    legend.title = element_text(size = 12, face = "bold")
  ) +
  labs(
    x = "",
    y = "",
    title = "Change in use of transit stations",
    color = "Region"
  )

timeseries_all_2
```



These graphs would need further work. We could for example divide the data between North, Centre and Southern Italy. Let's load the geojson of italian regions and plot the data.

```
# Add the polygons of Italy to the environment
italy_iso3166 <- st_read("data/longitudinal-1/italy_projected_simplified.geojson")
```

```
Reading layer `italy_projected_simplified' from data source
```

```
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202324/r4ps/data/longitudinal-
  using driver `GeoJSON'
Simple feature collection with 124 features and 4 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 1313364 ymin: 3933695 xmax: 2312062 ymax: 5220353
Projected CRS: Monte Mario / Italy zone 1
```

```r
# create a simple plot
plot(italy_iso3166$geometry)
```

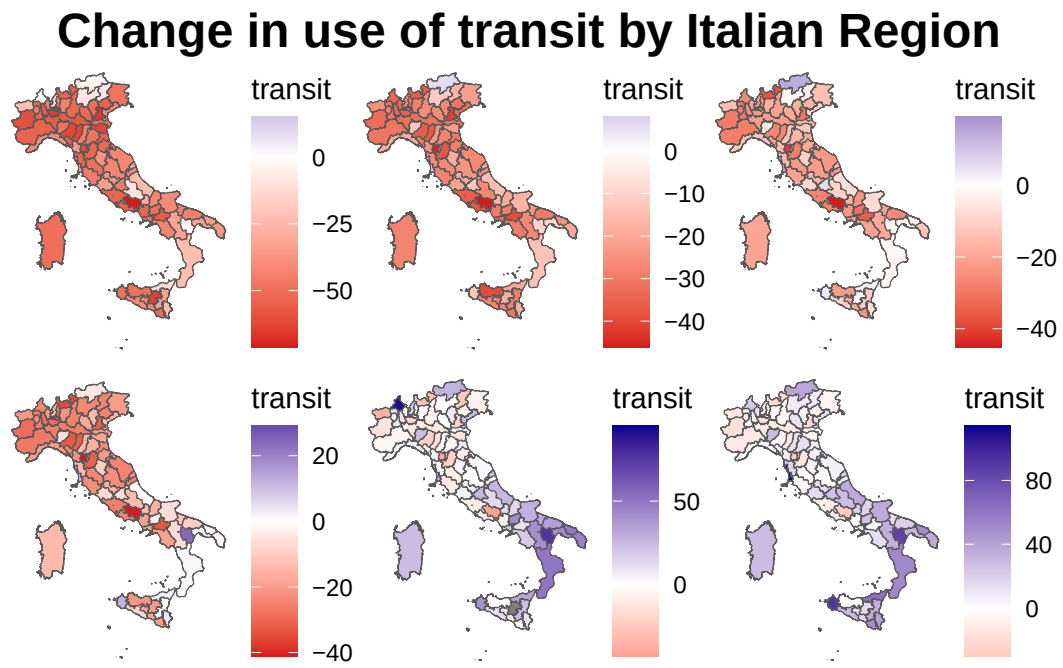
Now let's join the data by regional code.

```r
# Join by regional code
italy_iso3166_tidy <- italy_iso3166 %>%
  left_join(mobility_it, by=c("ISO3166.2"="iso_3166_2_code"))

# check the first rows of the merged data table
# head(italy_iso3166_tidy)
```

To start with, we can plot the data statically using ggplot.

```r
# One point in time
italy_sf_subset_1 <- italy_iso3166_tidy %>% filter(date_fix == "2022-01-01")
italy_sf_subset_2 <- italy_iso3166_tidy %>% filter(date_fix == "2022-02-01")
italy_sf_subset_3 <- italy_iso3166_tidy %>% filter(date_fix == "2022-03-01")
italy_sf_subset_4 <- italy_iso3166_tidy %>% filter(date_fix == "2022-04-01")
italy_sf_subset_5 <- italy_iso3166_tidy %>% filter(date_fix == "2022-05-01")
italy_sf_subset_6 <- italy_iso3166_tidy %>% filter(date_fix == "2022-06-01")

#This may take some time to load
g1 <- ggplot() +
  geom_sf(data = italy_sf_subset_1, aes(fill = transit, geometry = geometry)) +
  scale_fill_gradient2(low = "#d7191c", mid = "white", high = "darkblue",
                       midpoint = 0, guide = "colourbar") +
  theme_void()


g2 <- ggplot() +
  geom_sf(data = italy_sf_subset_2, aes(fill = transit, geometry = geometry)) +
    scale_fill_gradient2(low = "#d7191c", mid = "white", high = "darkblue",
                       midpoint = 0, guide = "colourbar") +
  theme_void()

g3 <- ggplot() +
  geom_sf(data = italy_sf_subset_3, aes(fill = transit, geometry = geometry)) +
    scale_fill_gradient2(low = "#d7191c", mid = "white", high = "darkblue",
                       midpoint = 0, guide = "colourbar") +
  theme_void()

g4 <- ggplot() +
  geom_sf(data = italy_sf_subset_4, aes(fill = transit, geometry = geometry)) +
    scale_fill_gradient2(low = "#d7191c", mid = "white", high = "darkblue",
                       midpoint = 0, guide = "colourbar") +
  theme_void()

g5 <- ggplot() +
  geom_sf(data = italy_sf_subset_5, aes(fill = transit, geometry = geometry)) +
    scale_fill_gradient2(low = "#d7191c", mid = "white", high = "darkblue",
                       midpoint = 0, guide = "colourbar") +
  theme_void()

g6 <- ggplot() +
```

```
  geom_sf(data = italy_sf_subset_6, aes(fill = transit, geometry = geometry)) +
    scale_fill_gradient2(low = "#d7191c", mid = "white", high = "darkblue",
                         midpoint = 0, guide = "colourbar") +
  theme_void()

grid.arrange(g1, g2, g3, g4, g5, g6, ncol = 3,
             top = textGrob("Change in use of transit by Italian Region", gp=gpar(fontsize
```



Plotting time and space interactively is more complex. There are quite a few examples of how to visualise data over space and time here.

8.5 Questions

For the assignment, we will continue to focus on the United Kingdom as our geographical area of analysis. For this section, we will use Google Mobility data for the UK for 2021. It has the same format as the Google Mobility data for Italy used in this chapter. During this year the UK underwent a third national lockdown, limitations of gatherings and more. For details on the timeline you can have a look here.

Start by loading both the csv and geojsons.

```r
mobility_gb <- read.csv("data/longitudinal-1/2021_GB_Region_Mobility_Report.csv", header =

uk_iso3166 <- st_read("data/longitudinal-1/uk_projected_simplified.geojson")
```

```
Reading layer `uk_projected_simplified' from data source
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202324/r4ps/data/longitudinal-
  using driver `GeoJSON'
Simple feature collection with 245 features and 4 fields (with 16 geometries empty)
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -50462.93 ymin: 5270.466 xmax: 655975.3 ymax: 1219809
Projected CRS: OSGB36 / British National Grid
```

1. Chose three variables to focus on and plot them together using `ggplot`. Discuss what format the data and if you can see any shocks over time by referring to the UK COVID-19 timeline.
2. Chose one variable and fit a linear, quadratic or spline model. Justify why you think this is the most appropriate and how you would interpret the resulting equation.
3. Model both time and space: use `ggplot` to create a graph of the data by region or other geographical area. You can chose what to focus on here and do not necessarily need to use all the geographical classifications available to you. Analyse the trends that appear in your plot.
4. Create a static or interactive map (bonus points) of your choice from the data.

Analyse and discuss what insights you obtain into people's attitudes, concerns, and behaviors during the pandemic, as well as their response to interventions and policies.

9 Assessing Interventions

Since its outbreak in late 2019, COVID-19 affected millions of people worldwide. One of the ways to track the spread of the disease is through the collection and analysis of data on daily cases, which has become a crucial component of new digital footprint trends. Contact tracing apps and tools were developed and implemented in many countries to help track the spread of COVID-19.

Data on daily cases can help us evaluate the effectiveness of policies implemented during the pandemic, such as lockdowns and other restrictions. By analyzing these data, policymakers and public health experts have gained insights into the impact of these policies on the spread of the disease Brodeur et al. (2021) and (Zhou and Kan 2021). For instance, if daily cases decrease significantly during a lockdown, it could indicate that the policy was successful in controlling the spread of the disease.

Data on daily cases can also help policymakers to make informed decisions about when to implement or lift certain restrictions. For example, if daily cases start to rise again after a lifting of restrictions, it may indicate that it is too early to do so, and that more caution is necessary. In summary, data on daily cases was an essential tool for evaluating the effectiveness of policies implemented during the COVID-19 pandemic and making informed decisions about future policies. As we will see in this chapter, evaluating COVI9-19 policies based onn cases is, unfortunately not so simple.

To assess government interventions during the pandemic, we will be using **difference-in-differences** (diff-n-diff), a widely-adopted quasi-experimental design, used to evaluate the effect of an intervention or exposure on an outcome of interest. This strategy involves comparing the change in the outcome before and after the intervention or exposure in the treated group, relative to the change in the outcome over the same time period in a control group. By doing so, diff-n-diff can help isolate the effect of the intervention or exposure from other factors that may be driving the outcome.

This chapter is based on :

- Goodman-Bacon, Andrew, and Jan Marcus. "Using difference-in-differences to identify causal effects of COVID-19 policies." (2020)
- Andrew Heiss' chapter on Difference-in-differences
- Brodeur, Abel, et al. "COVID-19, lockdowns and well-being: Evidence from Google Trends." Journal of public economics 193 (2021): 104346.

9.1 Dependencies

```
# Data
library(sf)
library(readr)
library(tidyverse)

# Dates and Times
library(lubridate)

# graphs
library(ggplot2)
library(ggthemes)
library(gridExtra)
library(viridis)
library(ggimage)
library(grid)
library(plotly)

# Models
library(modelsummary)
library(broom)
library(gtools)
```

9.2 Data

First let's import the Greater London COVID-19 data:

```
# import csv
covid_cases_london <- read.csv("data/longitudinal-2/covid_cases_london.csv", header = TRUE
# check out the variables
colnames(covid_cases_london)
```

```
[1] "areaType"                       "areaName"
[3] "areaCode"                       "date"
[5] "newCasesBySpecimenDate"         "cumCasesBySpecimenDate"
[7] "newFirstEpisodesBySpecimenDate" "cumFirstEpisodesBySpecimenDate"
[9] "newReinfectionsBySpecimenDate"  "cumReinfectionsBySpecimenDate"
```

Then we do the same for the Lazio area data, which is the Region of the capital of Italy, Rome. We are choosing this region because it did not see sharp peaks in COVID-19 cases during the winter of 2020/2021.

```
# import csv
covid_cases_lazio <- read.csv("data/longitudinal-2/covid_cases_lazio.csv", header = TRUE)
# check out the variables
colnames(covid_cases_lazio)
```

```
[1] "data"                    "stato"
[3] "codice_regione"          "denominazione_regione"
[5] "codice_provincia"        "denominazione_provincia"
[7] "sigla_provincia"         "lat"
[9] "long"                    "totale_casi"
```

First we need to clean up the data somewhat and rename some variables in both dataframes to have 4 variables:

- `date`: year-month-day
- `geo`: geographical region
- `cases`: number of COVID-19 cases that day
- `area` : Lazio (Rome) or London

```
# Rename the variables in the Lombardia data frame
covid_cases_lazio_ren <- covid_cases_lazio %>%
  rename(date = data , geo = denominazione_provincia, totalcases = totale_casi)

# Group the data by region and calculate total cases for each day (not cumulative cases)
covid_cases_lazio_daily <- covid_cases_lazio_ren %>%
  group_by(geo) %>%
   mutate(cases = totalcases - lag(totalcases, default = 0)) %>%
  select(date, geo, cases) %>%
  mutate(area = "Rome (Lazio)") %>%
  filter(cases >= 0)

#df_milan <- covid_cases_lombardia_new %>%
#  filter(geo == "Milano", new_cases >= 0)

# Rename the variables in the first data frame
covid_cases_london_ren <- covid_cases_london %>%
```

```
  rename(date = date , geo = areaName, cases = newCasesBySpecimenDate) %>%
  select(date, geo, cases) %>%
  mutate(area = "London")

# Correct date format
covid_cases_london_ren$date <- as.Date(covid_cases_london_ren$date)
covid_cases_lazio_daily$date <- as.Date(covid_cases_lazio_daily$date)

# Append the renamed data frame to the second data frame
covid_combined <- rbind(covid_cases_london_ren, covid_cases_lazio_daily)

# Add a variable of log of cases
covid_combined <- covid_combined %>%
          mutate(log_cases = log(cases))
```

9.3 Data Exploration

Similarly to the previous chapter. Let's start by eyeballing the data.

```
# Visualizing Cases in London
covid_cases_1 <- ggplot(data = covid_cases_london_ren, aes(x = date, y = cases, color=geo)
  geom_line() +
  scale_color_viridis(discrete = TRUE, option="magma") +
  theme_minimal() +
  theme(
    legend.position = "bottom")
  labs(
    x = "",
    y = "",
    title = "Evolution in Covid-19 cases",
    color = "Region"
  ) +
  scale_x_date(date_breaks = "6 months")
```

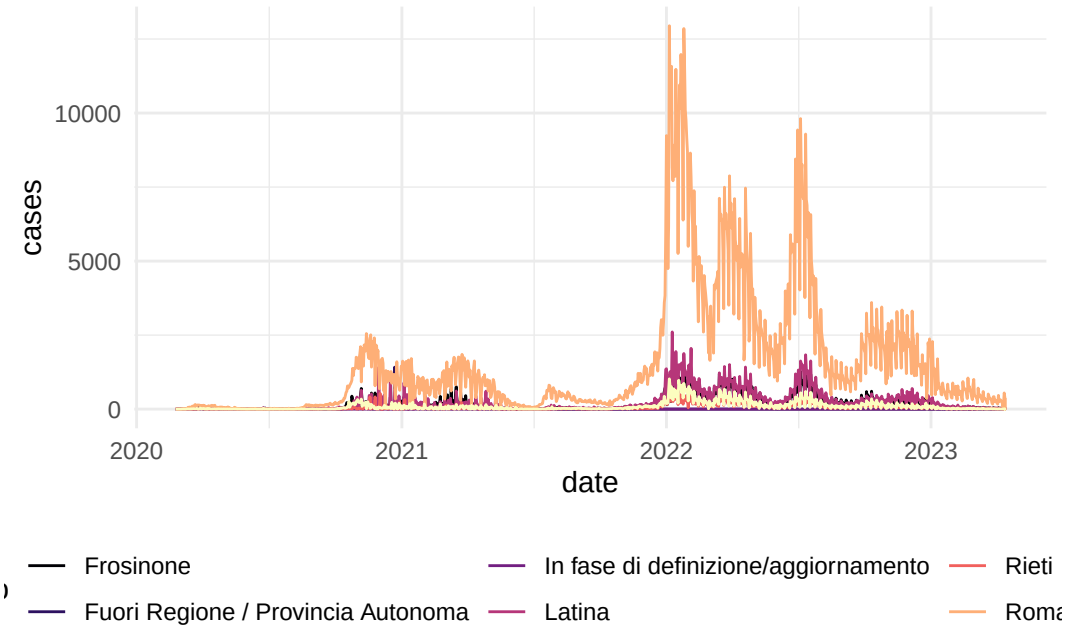
```
NULL
```

```
  covid_cases_1
```


```
# Visualizing Cases in Lazio (Rome)
covid_cases_2 <- ggplot(data = covid_cases_lazio_daily, aes(x = date, y = cases, color=geo
  geom_line() +
  scale_color_viridis(discrete = TRUE, option="magma") +
  theme_minimal() +
  theme(
    legend.position = "bottom")
  labs(
    x = "",
    y = "",
    title = "Evolution in Covid-19 cases",
    color = "Region"
  ) +
  scale_x_date(date_breaks = "6 months")
```

```
NULL
```

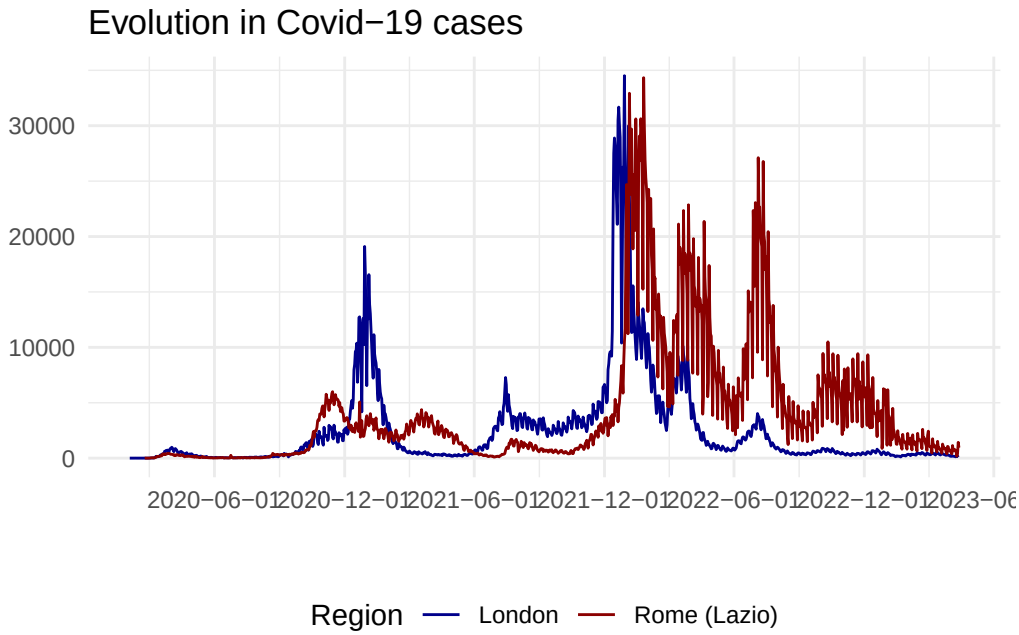
```
covid_cases_2
```



To identify whether there is a timeperiod where a lockdown was implemented in one location but not the other, and how cases evolved, we can plot aggregates of both locations in one plot.

```
# Aggregate the data by region for each day
covid_combined_agg <- aggregate(cases ~ area + date, data = covid_combined, FUN = sum)

# Visualizing aggregated
covid_cases_3 <- ggplot(data = covid_combined_agg, aes(x = date, y = cases, color=area)) +
  geom_line() +
  scale_color_manual(values=c("darkblue", "darkred")) + # set individual colors for the ar
  theme_minimal() +
  theme(
    legend.position = "bottom"
  ) +
  labs(
    x = "",
    y = "",
    title = "Evolution in Covid-19 cases",
    color = "Region"
  ) +
  scale_x_date(date_breaks = "6 months")
```

```
covid_cases_3
```

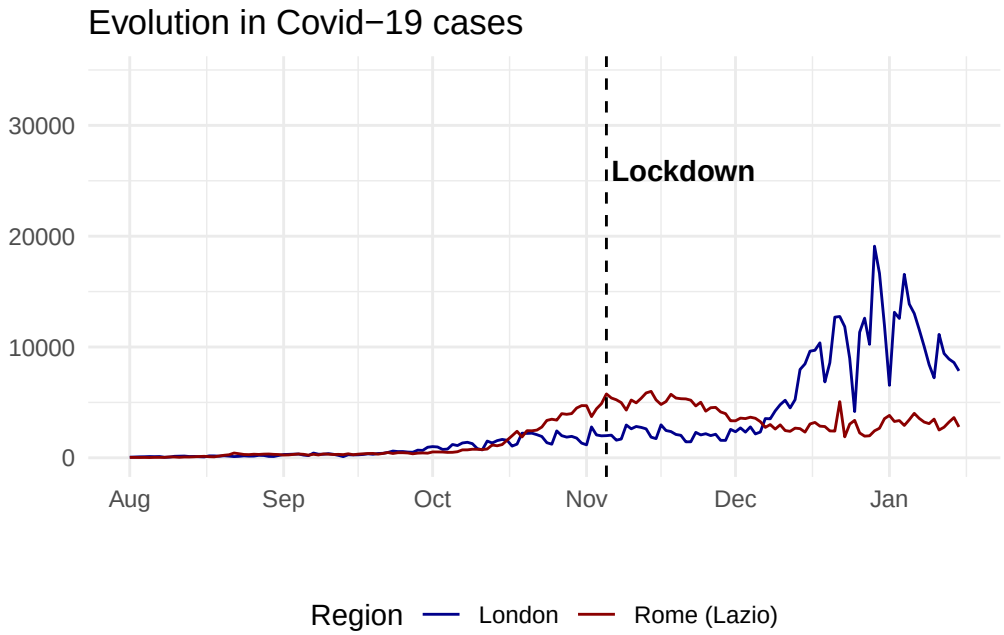


From an initial look at the data, the 2020/2021 winter period seems interesting as there is a high increase in London cases but not as much as a peak in Lazio cases. In fact, after a quick review of COVID-19 lockdowns, we found that:

- On the 5th of November 2020, the UK Prime Minister announced a second national lockdown, coming into force in England
- On 4 November 2020, Italian Prime Minister Conte announced a new lockdown as well, however this lockdown divided the country into three zones depending on the severity of the pandemic, corresponding to red, orange and yellow zones. The Lazio region, was a yellow zone for the duration of this second lockdown. In yellow zones, the only restrictions included compulsory closing for restaurant and bar activities at 6 PM, and online education for high schools only.

```
# Usual chart
covid_cases_4 <- ggplot(data = covid_combined_agg, aes(x = date, y = cases, color=area)) +
  geom_line() +
  scale_color_manual(values=c("darkblue", "darkred")) + # set individual colors for the ar
  theme_minimal() +
  theme(
```

```
    legend.position = "bottom"
  ) +
  labs(
    x = "",
    y = "",
    title = "Evolution in Covid-19 cases",
    color = "Region"
  ) +
  scale_x_date(limit=c(as.Date("2020-08-01"), as.Date("2021-01-15"))) +
 geom_vline(xintercept=as.numeric(as.Date("2020-11-05")), linetype="dashed") +
  annotate("text", x=as.Date("2020-11-06"), y=25000, label="Lockdown",
           color="black", fontface="bold", angle=0, hjust=0, vjust=0)

covid_cases_4
```



We could make some assumptions and set this up as a quasi experiment. In social science, researchers are often using natural or quasi experimental setting as randomized experiments can rarely be conducted. This involves splitting the population at hand into a treatment and control group.

9.4 Difference in Difference

Plotting Means

For a `diff-in-diff` analysis using COVID data, possible shocks that would make this type of quasi-experiment possible could be the following:

- **National lockdown**: The first national lockdown in the UK was announced on March 23, 2020. This sudden shock to the economy and society could be used as a treatment group for the diff-in-diff analysis, with the pre-lockdown period as the control group.
- **Regional lockdowns**: The UK also implemented regional lockdowns throughout the pandemic, with different regions experiencing restrictions at different times. These regional lockdowns could be used as treatment groups, with regions that did not experience lockdowns as the control group.
- **School closures**: In response to the pandemic, schools in the UK were closed from March 20, 2020, until June 1, 2020, and then again from January 5, 2021, until March 8, 2021. The impact of school closures on education outcomes could be studied using a diff-in-diff approach, with the period before school closures as the control group.
- **Travel restrictions**: The UK implemented various travel restrictions throughout the pandemic, including quarantine requirements for travelers from certain countries. The impact of these travel restrictions on the tourism industry or the spread of the virus could be studied using a diff-in-diff approach.
- **Vaccine rollout**: The UK began its COVID-19 vaccination program in December 2020. The impact of the vaccine rollout on various health and economic outcomes could be studied using a diff-in-diff approach, with the period before the rollout as the control group.

These are just a few examples of shocks that could be used for a diff-in-diff analysis using COVID data. The choice of shock will depend on the research question and the data available.

The DiD approach includes a before-after comparison for a treatment and control group. In our example:

- A `cross-sectional comparison` (= compare a sample that was treated (London) to an non-treated control group (Rome))
- A `before-after comparison` (= compare treatment group with itself, before and after the treatment (5th of November))

The **main assumption** is that without the change in the natural environment the outcome variable would have remained constant!

First, we create a dummy variable to indicate the time when the treatment started. In our case this will be the 5th of November 2020. We will also limit the time-span of our data.

```
# keep data from 2020-09-01 to 2021-01-01
covid_combined_filtered <- covid_combined %>%
  filter(date >= "2020-09-01" & date <= "2021-01-01")

# create a dummy variable to indicate the time when the treatment started (5 Nov 2020)
covid_combined_filtered <- covid_combined_filtered %>%
  mutate(after_5nov = ifelse(date >= "2020-11-05", 1, 0)) #changed to 05 Nov
```

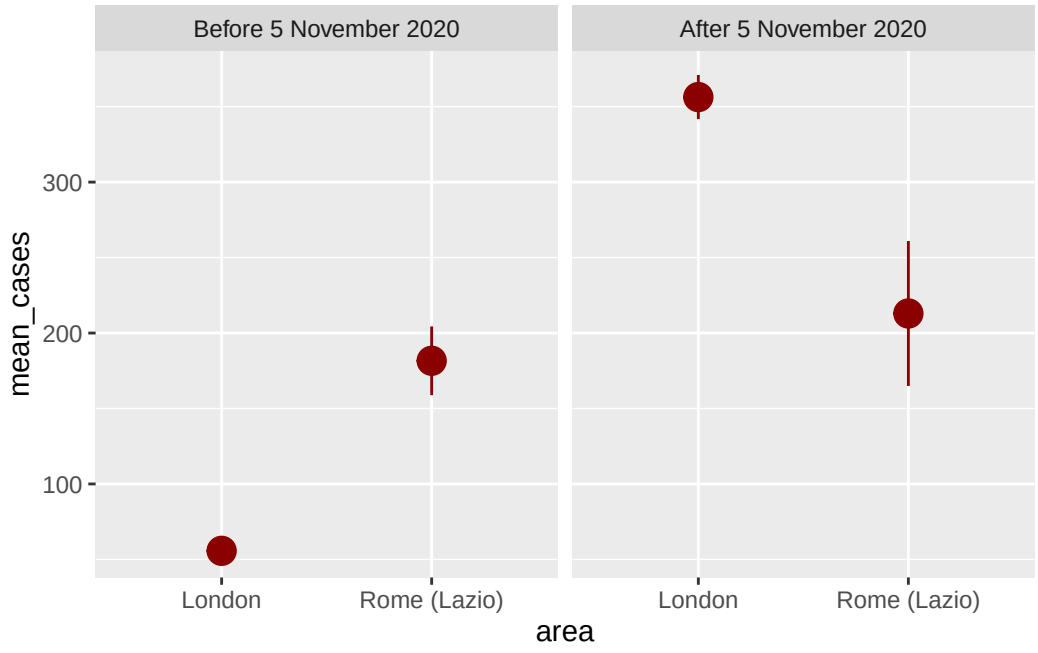
```
# Create a frequency table of area and treatment
freq_table <- table(covid_combined_filtered$area, covid_combined_filtered$after_5nov)

# Print the frequency table
print(freq_table)
```

```
               0    1
London       3150  540
Rome (Lazio) 1436  240
```

We then want to plot averages to see differences between treatment/control groups and before/after. But we can also calculate the mean and 95% confidence interval. We can also use group_by() and summarize() to figure out group means before sending the data to ggplot.

```
plot_data <- covid_combined_filtered %>%
  # Make these categories instead of 0/1 numbers so they look nicer in the plot
  mutate(after_5nov = factor(after_5nov, labels = c("Before 5 November 2020", "After 5 Nov
  group_by(area, after_5nov) %>%
  summarize(mean_cases = mean(cases),
            se_cases = sd(cases) / sqrt(n()),
            upper = mean_cases + (1.96 * se_cases),
            lower = mean_cases + (-1.96 * se_cases))

ggplot(plot_data, aes(x = area, y = mean_cases)) +
  geom_pointrange(aes(ymin = lower, ymax = upper),
                  color = "darkred", size = 1) +
```

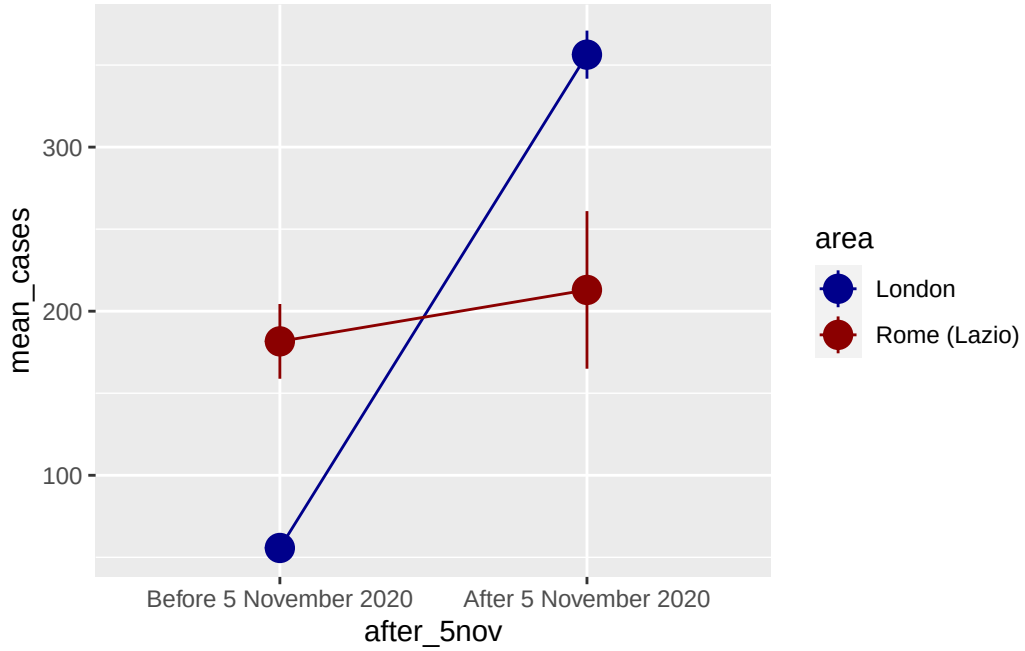
```
  facet_wrap(vars(after_5nov))
```



Here, we can start to see a diff-in-diff plot, where there is little to no difference in means with our control city (Rome-Lazio) and a substancial jump in means in our treatment city (London). It looks there were many more cases of COVID-19 after the 5th of march in London, indicating the lockdown did not have an effect, at least in this time-frame. Why could that be?

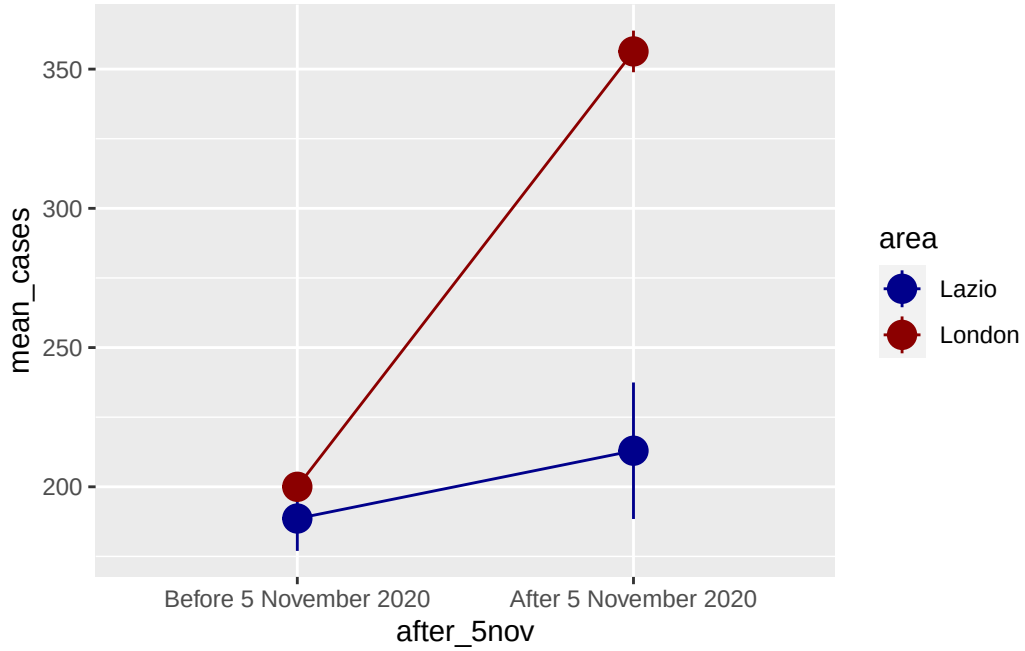
We can also plot a more standard diff-in-diff format:

```
ggplot(plot_data, aes(x = after_5nov, y = mean_cases, color = area)) +
  geom_pointrange(aes(ymin = lower, ymax = upper), size = 1) +
  geom_line(aes(group = area)) +
  scale_color_manual(values = c("darkblue", "darkred"))
```



This second plot shows us it is probable that our diff-n-diff set up will not work. A clean classic diff-n-diff would look more like the following. Please note the following plot is theoretical.

```
# import csv
covid_perfect_example <- read_csv("data/longitudinal-2/example_covid.csv")

# label pre/post labels
covid_perfect_example <- covid_perfect_example %>%
  mutate(after_5nov = factor(after_5nov, labels = c("Before 5 November 2020", "After 5 Nov

# plot
ggplot(covid_perfect_example, aes(x = after_5nov, y = mean_cases, color = area)) +
  geom_pointrange(aes(ymin = lower, ymax = upper), size = 1) +
  geom_line(aes(group = area)) +
  scale_color_manual(values = c("darkblue", "darkred"))
```



Difference in Difference by hand

We can find the exact difference by filling out the 2x2 before/after treatment/control table:

	Before	After	Difference
Treatment	A	B	B - A
Control	C	D	D - C
Difference	C - A	D - B	(D − C) − (B − A)

A combination of group_by() and summarize() makes this really easy. We can pull each of these numbers out of the table with some filter()s and pull():

```
before_treatment <- covid_perfect_example %>%
  filter(after_5nov == "Before 5 November 2020", area == "London") %>%
  pull(mean_cases)

before_control <- covid_perfect_example %>%
  filter(after_5nov == "Before 5 November 2020", area == "Lazio") %>%
  pull(mean_cases)

after_treatment <- covid_perfect_example %>%
```

```
  filter(after_5nov == "After 5 November 2020", area == "London") %>%
  pull(mean_cases)

after_control <- covid_perfect_example %>%
  filter(after_5nov == "After 5 November 2020", area == "Lazio") %>%
  pull(mean_cases)

diff_treatment_before_after <- after_treatment - before_treatment
diff_treatment_before_after
```

```
[1] 156.35
```

```
diff_control_before_after <- after_control - before_control
diff_control_before_after
```

```
[1] 24.35387
```

```
diff_diff <- diff_treatment_before_after - diff_control_before_after
diff_diff
```

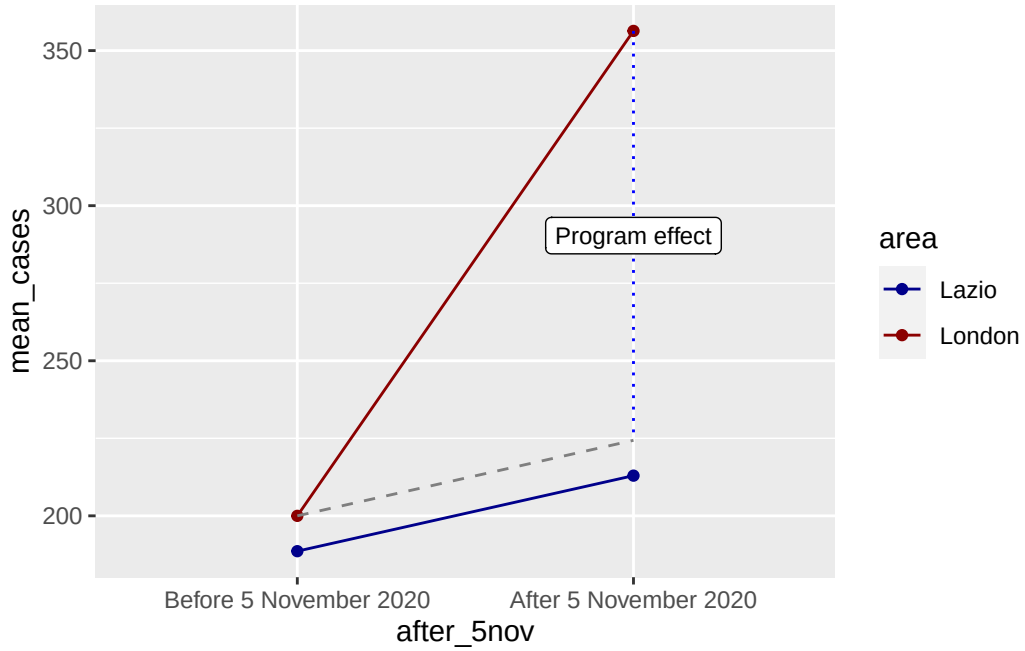
```
[1] 131.9961
```

The diff-in-diff estimate is 131.99, which means that the lockdown here caused an increase in cases in the time-window we are analysing. Not it's intended effect!

We can visualise this really well with a bit of extra code:

```
ggplot(covid_perfect_example, aes(x = after_5nov, y = mean_cases, color = area)) +
  geom_point() +
  #geom_pointrange(aes(ymin = lower, ymax = upper), size = 1) +
  #geom_line(aes(group = area)) +
  geom_line(aes(group = as.factor(area))) +
  scale_color_manual(values = c("darkblue", "darkred")) +
  # If you use these lines you'll get some extra annotation lines and
  # labels. The annotate() function lets you put stuff on a ggplot that's not
  # part of a dataset. Normally with geom_line, geom_point, etc., you have to
  # plot data that is in columns. With annotate() you can specify your own x and
  # y values.
  annotate(geom = "segment", x = "Before 5 November 2020", xend = "After 5 November 2020",
```

```
          y = before_treatment, yend = after_treatment - diff_diff,
          linetype = "dashed", color = "grey50") +
annotate(geom = "segment", x = "After 5 November 2020", xend = "After 5 November 2020",
          y = after_treatment, yend = after_treatment - diff_diff,
          linetype = "dotted", color = "blue") +
annotate(geom = "label", x = "After 5 November 2020", y = after_treatment - (diff_diff /
          label = "Program effect", size = 3)
```



It is important for all diff-in-diff analyses to give careful attention to possible violations of the common trends assumption, especially considering the COVID-19 situation where many of these violations are likely to occur. Furthermore, due to the unique dynamics of COVID-19 such as lags between exposure and recorded infections, nonlinearities from person-to-person transmission, and the possibility of policies having differential effects over time, it further complicates the potential risks to the diff-in-diff research design.

(Goodman-Bacon and Marcus 2020) comment on the following problems which can be consulted in their paper:

- Packaged Policies
- Reverse Causality
- Voluntary Precautions

- Difference Data collection
- Anticipation Spillovers
- Variation in Policy Timing

(Goodman-Bacon and Marcus 2020) also give great recommendations on how to address these problems, but this is far beyond the objective of this chapter.

Difference-in-Difference with regression

Calculating all the pieces by hand like that is tedious, so we can use regression to do it instead! Remember that we need to include indicator variables for treatment/control and for before/after, as well as the interaction of the two.

This is the equation:

$$\Delta Y_{gt} = \beta_0 + \beta_1 London_g + \beta_2 Post5Nov_t + \beta_3 London_g \times Post5Nov_t + \beta_4 Rome_g + \epsilon_{gt}$$

The output will show the diff-in-diff coefficient estimate, standard error, t-value, and p-value, which can be used to determine whether there was a significant effect of the second lockdown 4 on Covid cases in November 2020.

```
model_small <- lm(cases ~ area + after_5nov + area * after_5nov,
                  data = covid_combined_filtered)

# Tidy the model output
diffndiff1 <- tidy(model_small)

# Add significance stars using stars.pval from gtools
diffndiff1$stars <- stars.pval(diffndiff1$p.value)

# View the results
diffndiff1
```

```
# A tibble: 4 x 6
  term                        estimate std.error statistic   p.value stars
  <chr>                          <dbl>     <dbl>     <dbl>     <dbl> <chr>
1 (Intercept)                     55.7      4.47      12.5 3.62e- 35 ***
2 areaRome (Lazio)               126.       7.99      15.8 8.81e- 55 ***
3 after_5nov                     301.      11.7       25.7 7.63e-138 ***
4 areaRome (Lazio):after_5nov   -269.      21.0      -12.8 5.37e- 37 ***
```

```
# Create a model summary table for the model
summary_table <- modelsummary(list("Simple" = model_small), estimate =c("{estimate}{stars}
```

	Simple
(Intercept)	55.694***
	(4.469)
areaRome (Lazio)	125.910***
	(7.986)
after_5nov	300.656***
	(11.681)
areaRome (Lazio) × after_5nov	−269.302***
	(21.032)
Num.Obs.	5366
R2	0.130
R2 Adj.	0.130
AIC	74 524.8
BIC	74 557.7
Log.Lik.	−37 257.375
F	267.482
RMSE	250.71

```
# View the results
summary_table
```

9.5 Questions

For the assignment, you will continue to use Google Mobility data for the UK for 2021. For details on the timeline you can have a look here. You will need to do a bit of digging on when lockdowns or other COVID-19 related shock happened in 2021 to set up a diff-in-diff strategy. Have a look at Brodeur et al. (2021) to get some inspiration. They used Google Trends data to test whether COVID-19 and the associated lockdowns implemented in Europe and America led to changes in well-being.

Start by loading both the csv

```
mobility_gb <- read.csv("data/longitudinal-1/2021_GB_Region_Mobility_Report.csv", header =
```

1. Visualize the data with `ggplot` and identify what section of the data could be used to evaluate a COVID-19 intervention. Examples of these interventions could be a regional lockdown, school closures, travel restrictions or vaccine rollouts. Generate a clean `ggplot` which indicates which intervention you are going to examine.

2. Explore differences in means through a `frequency table` and a graph of these averages. Chose whichever suits your purposes best.
3. Define and estimage a diff-in-diff regression. What do the results suggest? Was the intervention you chose effective? Discuss the reasons why it was or was not.
4. Discuss how the unique dynamics of COVID-19 and the possibility of policies having differential effects over time complicate the interpretation of your results.

Analyse and discuss what insights you obtain into people's changes in behaviors during the pandemic in responde to an intervention.

10 Machine Learning

Machine learning (ML) is one of the most wanted skills in today's job market. It has very wide applications in many areas, from developing software for self-driving cars to early detection of deadly diseases such as cancer, but also in the studies of populations and human geography. Although the term **machine learning** has become a lot more popular in the last few years thanks to the increased capabilities to analyse larger amounts of data, many ML techniques have existed for decades in other fields such as statistics and computer science.

Just like in regression problems, ML algorithms build models based on sample data in order to make predictions or decisions about other unseen data. In fact, linear regression can be regarded as a very simple case of a neural network. There are different approaches to ML according to the type of feedback available to the ML algorithms:

- Supervised learning: we give the algorithm data where the inputs and outputs are already mapped, then the algorithm learns patterns of this mapping to be applied on new unseen data sets. Linear regression is an example of supervised learning.
- Unsupervised learning: we give the algorithm data where the inputs have no matched outputs. The task of the algorithm is to learn patterns in the data or processes to achieve a certain goal. An example of this is the k-means clustering algorithm that you explored in previous chapters.
- Reinforcement learning: the algorithm must achieve a goal while interacting with a dynamic environment. While navigating the environment, the algorithm is provided with feedback in the form of rewards which it tries to maximise (Bishop 2006). An example of this would be some of the software involved in self-driving cars.

In this chapter we will focus only on two ML techniques which are, however, very versatile since they can be used for classification, regression and other tasks. The first and most basic of the techniques that we will explore is **decision trees**. Building on these, we will then explore **random forests**. Both methods belong to the supervised learning approach. To illustrate the usage of these techniques within the context of population science, we will look at predicting the annual household income across the UK geography based on demographic data obtained from the census.

This chapter is based, among others, on the following references:

- Hands-on Machine Learning with R. Chapter 9: Decision trees (Boehmke 2019).

- UC Business Analytics R Programming Guide, developed by Brad Boehmke. Specifically, the chapters on Regression Trees & Bagging and Random Forests, which can be found here.

10.1 Dependencies

```
# Import the dplyr package for data manipulation
suppressMessages(library(dplyr))
# Import the rpart package for decision tree modeling
suppressMessages(library(rpart))
# Import the rpart.plot package for visualization of decision trees
suppressMessages(library(rpart.plot))
# Import ggplot 2 to make plots
suppressMessages(library(ggplot2))
# Import Metrics for performance metrics calculation
suppressMessages(library(Metrics))
# Import caret for machine learning modeling and evaluation
suppressMessages(library(caret))
# Import randomForest for the random forest algorithm
suppressMessages(library(randomForest))
# Import ranger for the ranger implementation of random forest, which is optimised for per
suppressMessages(library(ranger))
```

10.2 Data

As mentioned above, we will learn about decision trees and random forests through a practical example where the goal is to predict the net annual household income across different regions of the UK. Given that UK censuses take place separately in Northern Ireland, Scotland and England & Wales, we will focus only on England & Wales for simplicity.

In the code chunk below, we load a data set that has already been prepared for this notebook. It includes a variety of variables related to demographic characteristics of the population, aggregated at the Middle-Layer Super Output Area (MSOA) level. All the variables, except for the average annual household income, are derived from the 2021 census and the raw data can be downloaded from here. The annual household income data is from the Annual Survey of Hours and Earnings and can be downloaded from this link.

```r
# Load the data
df_MSOA <- read.csv("./data/machine-learning/census2021-msoa-income.csv")
# Data cleaning, remove the X field
df_MSOA$X <- NULL
# Data cleaning, the fields "date", "geography" and "geography.code" are not needed
df_MSOA <- subset(df_MSOA, select = -c(date, geography, geography.code))
# More data cleaning, remove comma from income and turn it into a numeric value
df_MSOA$INCOME <- as.numeric(gsub(",", "", df_MSOA$INCOME))
```

For a description of the variables in the columns of `df_MSOA`, we can load a dictionary for these variables:

```r
# Load variable dictionary
df_dictionary <- read.csv("./data/machine-learning/Dictionary.csv")
head(df_dictionary)
```

```
                                    Dictionary       X
1
2                                         Name     Key
3                 Lives in household (% persons)    inHH
4    Lives in communal establishment (% persons)    inCE
5 Never married or civil partnership (% persons)    SING
6    Married or in civil partnership (% persons) MARRIED
```

10.3 Splitting the data

For most supervised learning problems, the goal is to find an algorithm or model that not only fits well known data, but also accurately predicts unknown values of the average annual household income based on a set of inputs. In other words, we want the algorithm to be generalisable. In order to measure the generalisability of the optimal algorithm, we can split the data into a training set containing input and output data and a test set. The training set is used to teach the algorithm how to map inputs to outputs, and the test set is used to estimate the prediction error of the final algorithm, which quantifies the generalisability of the model. The test set should never be used during the training stage.

As a rule of thumb, the data should be split so that 70% of the samples are in the training set and 30% in the test set, although this percentages might vary slightly according to the size of the original data set. Furthermore, to ensure generalisability, the data split should be done so that the distribution of outputs in both the training and test set is approximately the same.

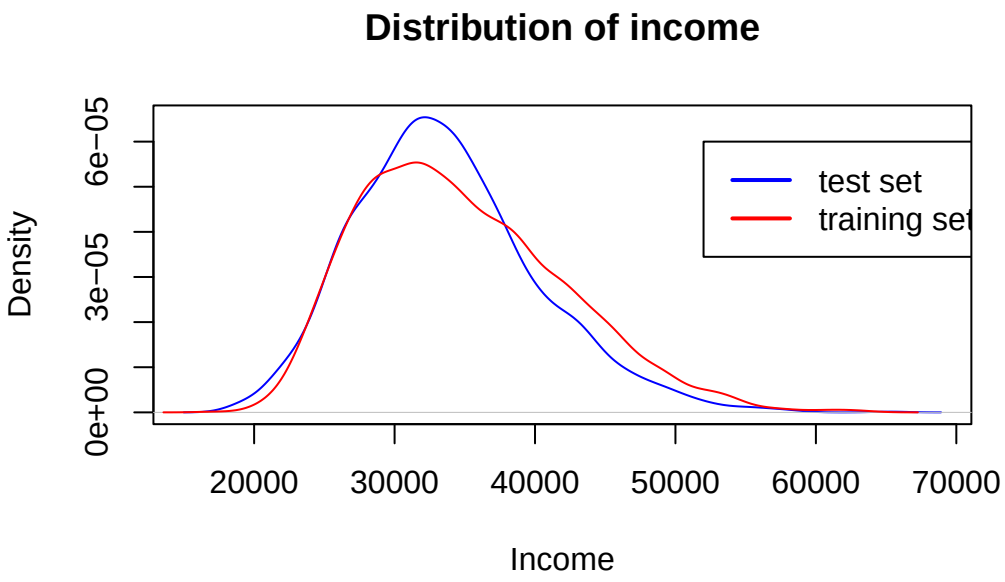
The function `create_train_test` below (borrowed from here) allows us to select samples from the data to create the training set (when the train parameter is set to `TRUE`) and the test set (when the train parameter is set to `FALSE`).

```
create_train_test <- function(data, size = 0.7, train = TRUE) {
 n_row = nrow(data)
 total_row = size * n_row
 train_sample <- 1: total_row
 if (train == TRUE) {
 return (data[train_sample, ])
 } else {
 return (data[-train_sample, ])
 }
}
```

```
df_train <- create_train_test(df_MSOA, 0.7, train = TRUE)
df_test <- create_train_test(df_MSOA, 0.7, train = FALSE)
```

If the data is naively split into training and test sets as we did above, the distribution of outputs in the training and test set will not be the same, as the following plot shows.

```
d1 <- density(df_test$INCOME)
plot(d1, col='blue', xlab="Income", main="Distribution of income")

d2 <- density(df_train$INCOME)
lines(d2, col='red')

legend(52000, 0.00006, c("test set","training set"), lwd=c(2,2), col=c("blue","red"))
```



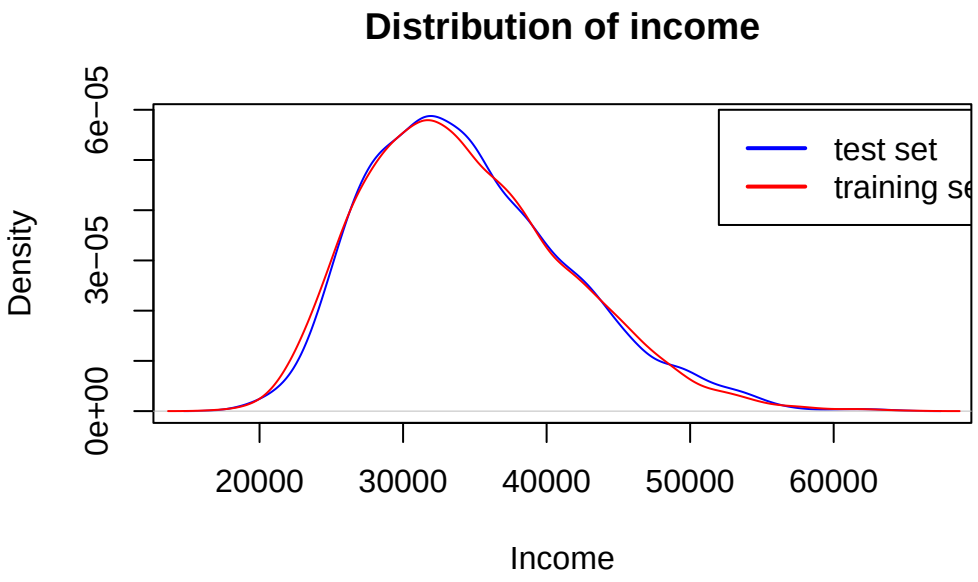
This is due to the fact that the dataset that we loaded at the beginning of this workbook is sorted so that some entries corresponding to MSOAs that are geographically close to each other are in consecutive rows. Therefore, to ensure that the distribution of outputs in training and test sets is the same, the data needs to be randomly shuffled. The code below achieves this goal, as it can be seen in the kernel density plot with the new data split.

```
# Shuffle the entries of the original dataset
shuffle_index <- sample(1:nrow(df_MSOA))
df_MSOA <- df_MSOA[shuffle_index, ]
```

```
#Perform the data split with the new shuffled dataset
df_train <- create_train_test(df_MSOA, 0.7, train = TRUE)
df_test <- create_train_test(df_MSOA, 0.7, train = FALSE)
```

```
#Plot the kernel density for both training and test data sets
d1 <- density(df_test$INCOME)
plot(d1, col='blue', xlab="Income", main="Distribution of income")

d2 <- density(df_train$INCOME)
lines(d2, col='red')
```

```
legend(52000, 0.00006, c("test set","training set"), lwd=c(2,2), col=c("blue","red"))
```



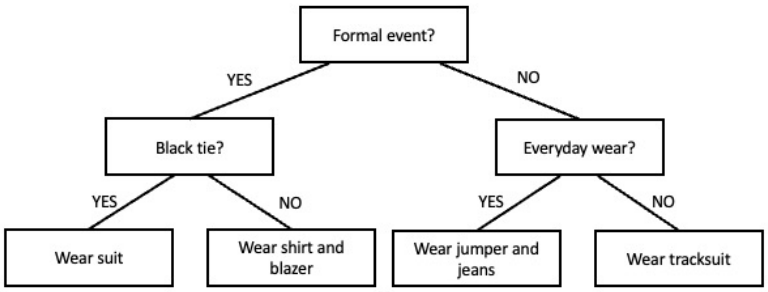
Before proceeding any further we should note here that one of the advantages of decision trees and random forests is that they are not sensitive to the magnitude of the input variables, so standardisation is not needed before fitting these models. However, there are other ML algorithms such as k-means, where standardisation is a crucial step to ensure the success of learning process and it should always take place before training the algorithm. Similarly, random forests are not sensitive to correlations between independent variables, so there is no need to check for correlations before training the models.

10.4 Decision trees

10.4.1 Fitting the training data

Decision trees are an ML algorithm capable of performing both classification and regression tasks, although in this workbook we will focus only on regression. One of their most notable advantages is that they are very interpretable, although their predictions are not always accurate. However, by aggregating decision trees through a method called bagging, the algorithm can become much more powerful.

In essence, a decision tree is like a flowchart that helps us make a decision based on a number of questions or conditions, which are represented by internal nodes. It starts with a root node and branches out into different paths, with each branch representing a decision. The final nodes represent the outcome and are known as leaf nodes. For example, imagine you are trying to decide what to wear to an event. You could create a decision tree with the root node being "Formal event?", like in the diagram below. If the answer is "yes" you would proceed to the left and if the answer if "no", you would proceed to the right. On the right, you would have another node for "Black tie?", and again two "yes" and "no" branches emerging from it. On the left, you would have a node for "Everyday wear?" with its two branches. Each branch would eventually lead to a decision or action represented by the so-called leaf nodes, such as "wear suit".



The decision tree to predict the annual household income of an MSOA based on its demographic features will be a lot more complex than the one depicted above. The branches will lead to leaf nodes representing the predicted values of annual household income. The internal nodes will represent conditions for the demographic variables (e.g. is the percentage of people aged 65 or over in this MSOA greater than X %?). Not all the demographic variables will be equally relevant to predict the annual household income at the MSOA level. To optimise the prediction process, conditions on the most relevant variables should be near the root of the tree so the most determining questions can be asked at the beginning of the decision-making process. The internal nodes that are further from the tree will help fine-tune the final predictions. But, how can we choose which are the most relevant variables? And, what are the right questions to ask in each internal node (e.g. if we are asking 'is the percentage of people aged 65 or over in this MSOA greater than X %?', what should be the value of X ?).

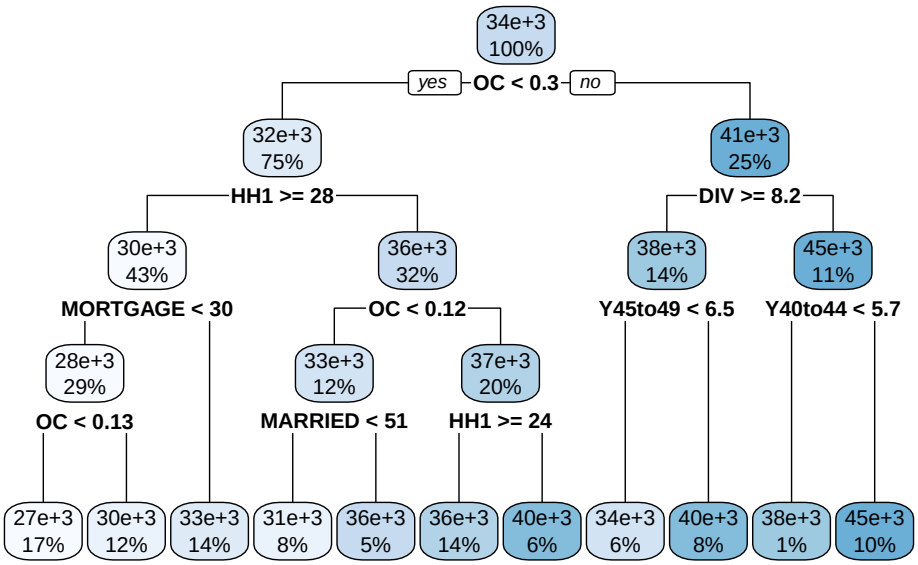
Luckily, nowadays there are many off-the-shelf software packages available that can help us build decision trees with just a line of code. Here, we use the R package `rpart`, which is based on the Classification And Regression Tree (CART) algorithm proposed by Breiman (Breiman

1984). In particular, we can use the function `rpart()` to find the decision tree that best fits the training set. This function requires to use the formula method for expressing the model. In this case, the formula is `INCOME ~.`, which means that we regard the variable `INCOME` as a function of all the other variables in the training data set. For more information on the formula method, you can check the R documentation. The function `rpart()` also requires to specify the method for fitting. Since we are performing a regression task (as opposed to classification), we need to set method to `'anova'`.

```
fit <- rpart(INCOME ~., data = df_train, method = 'anova')
```

We can visualise the fitted decision tree with the `rpart.plot()` function from the library with the same name. Interestingly, the condition for the root node is associated with the variable representing the percentage of people born in Antarctica, Oceania and Other, so if an MSOA has less than 0.25% people born in those territories, the model predicts it to have lower annual household income. Can you think of why this might be the case?

```
rpart.plot(fit)
```

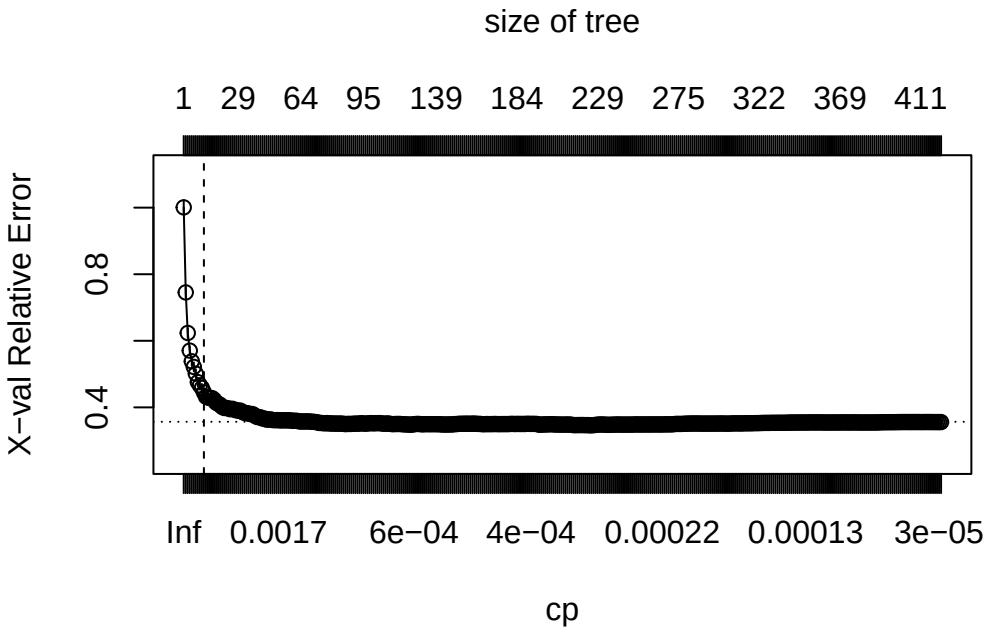


As we can see, `rpart()` produces a decision tree with 11 leaf nodes and the conditions to reach each leaf are associated with only 8 demographic variables. However, in the original training set, there were many more demographic variables that have not been included in the model. The reason for this is that, behind the scenes, `rpart()` is trying to find a balance between

the complexity of the tree (i.e. its depth) and the generalisability of the tree to predict new unseen data. If it is too deep, the tree runs the risk of overfitting the training data and failing to predict the annual household income for other MSOAs that are not included in the training set.

To illustrate the point of selecting a tree with 11 leaves, we can manually control the level of complexity allowed when fitting of a decision tree model. The lower we set the value of the parameter `cp`, the more complex the resulting tree will be, so `cp` can be regarded as penalty for the level of complexity or a cost complexity parameter. Below, we fit a decision tree with no penalty for generating a complex tree model, i.e. `cp=0`, and then we use the function `plotcp()` to plot the prediction error (PRESS statistic) that would be achieved with decision trees of different levels of complexity.

```
# set seed for reproducibility
set.seed(123)
# train model
fit2 <- rpart(INCOME ~., data = df_train, method = 'anova', control = list(cp = 0))
# plot error vs tree complexity
plotcp(fit2)
# draw vertical line at 11 trees
abline(v=11, lty='dashed')
```



As we can see from the plot above, with more than 11 leaves, little reduction in the prediction

error is achieved as the model becomes more and more complex. In other words, we start seeing diminishing returns in error reduction as the tree grows deeper. Hence `rpart()` is doing some behind-the-scenes tuning by pruning the tree so it only has 11 leaf nodes.

There are other model parameters that can be tuned in order to improve the model performance via the control argument of the `rpart()` function, just like we did above for `cp`. While we do not experiment with these additional parameters in the workbook, we provide brief descriptions below so you can explore them on your own time:

- `minsplit` controls the minimum number of data points required in each leaf node. The default is 20. Setting this lower will result in more leaves with very few data points belonging to the corresponding branch.
- `maxdepth` controls the maximum number of internal nodes between the root node and the terminal nodes. By default, it is set to 30. Setting it higher allows to create deeper trees.

10.4.2 Measuring the performance of regression models

To measure the performance of the regression tree that we fitted above, we can use the test set. We firstly use the `predict()` function from the `rpart` library to compute some predictions on the MSOA annual household income for the test set data.

```
predict_unseen <-predict(fit, df_test)
```

Then, we compare the predictions with the actual values and measure the discrepancy with a regression error metric.

Note that, **to measure the performance of classification trees, the procedure would be slightly different** and it would involve the computation of a confusion matrix and the accuracy metric.

Different error metrics exist to measure the performance of regression models such as the Mean Squared Error (MSE), the Mean Absolute Error (MAE) or the Root Mean Squared Error (RMSE). The MSE is more sensitive to outliers than the MAE, however, the units of MSE are squared units. The RMSE solves the problem of the squared units associated with MSE by taking its squared root. The library `Metrics` provides the in-built function `rmse()` which makes the computation of RMSE straightforward:

```
rmse(predict_unseen, df_test$INCOME)
```

```
[1] 4747.223
```

This value does not have much meaning in isolation. A good or bad RMSE is always relative to the specific data set. For this reason, we need to establish a baseline RMSE that we can compare it with. Here we establish this baseline as the RMSE that would be obtained from a naive tree that merely predicts the mean annual household income value across all the data entries in the training set. If the fitted model achieves an RSME lower than the naive model, we say that the fitted model "has skill". The following line of code confirms that our fitted model is better than the naive model.

```
rmse(predict_unseen, mean(df_train$INCOME))
```

```
[1] 5290.561
```

10.4.3 Bagging

Even though single decision trees have many advantages such as being very simple and interpretable, their predictions are not always accurate due to their high variance. This results in unstable predictions that may be dependent on the chosen training data.

A method called bagging can help solve this issue by combining and averaging the predictions of multiple decision tree models. The method can actually be applied to any regression or classification model, however, it is most effective when applied to models that have high variance. Bagging works by following three steps:

1. Create m bootstrap samples from the training data (i.e. m random samples with replacement).
2. For each bootstrap sample, train an unpruned single tree (i.e. with `cp=0`).
3. To create a prediction for a new data point, input the data in the single trees fitted with each bootstrap sample. The prediction will be the average of all the individual predictions output by each tree.

Each bootstrap sample typically contains about two-thirds of the training data, leaving one-third out. This left-out third is known as the out-of-bag (OOB) sample and it provides a natural opportunity to cross-validate the predictive performance of the model. By cross-validating the model, we can estimate how well our model will perform on new data without necessarily having to use the test set to test it. In other words, we can use the whole of the original data set for training and still quantify the performance of the model. However, for simplicity, we will train the model on the training set only here.

Bagging can be easily done with a library called `caret`. Here we fit a 10-fold cross-validated model, meaning that the bagging is applied so that there are 10 different OOB samples.

```
# specify 10-fold cross validation
ctrl <- trainControl(method = "cv",  number = 10)
# set seed for reproducibility
set.seed(123)
# train the cross-validated bagged model
bagged <- caret::train(INCOME ~ ., data = df_train, method = "treebag", trControl = ctrl,

print(bagged)
```

```
Bagged CART

4956 samples
  48 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4460, 4459, 4460, 4462, 4461, 4461, ...
Resampling results:

  RMSE      Rsquared   MAE
  4185.099  0.6560846  3169.149
```

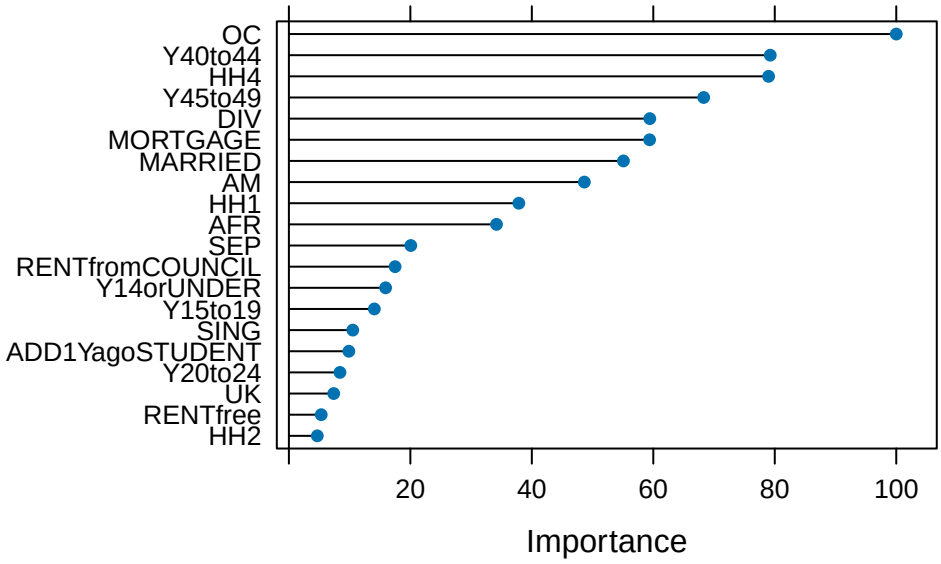
We see that the cross-validated value of RMSE with bagging is lower than that associated with the single decision tree that we trained with `rcart`. This indicates that the predictive performance is estimated to be better. We can compare the cross-validated value of the RMSE with the RMSE from the test set. These two quantities should be close:

```
predict_bagged_unseen <- predict(bagged, df_test)
rmse(predict_bagged_unseen, df_test$INCOME)
```

```
[1] 4225.871
```

The library `caret` has an additional function `varImp()` that helps us understand the variable importance across the bagged trees, i.e. the variables that are most relevant to determine the predictions of MSOA net annual household income. You are welcome to check the `caret` documentation to learn more about how the variable importance is determined. We can plot a rank of variable importance by running the code below.

```
plot(varImp(bagged), 20)
```



As noted before, given the variables included in our original data set, the percentage of people born in Antarctica, Oceania and Other is, almost invariably, the best predictor of annual household income, although due to the randomness introduced in bagging, this could sometimes change.

10.5 Random forests

While bagging considerably improves the performance of decision trees, the resulting models are still subject to some issues. Mainly, the multiple trees that are fitted through the bagging process are not completely independent of each other since all the original variables are considered at every split in every tree. As a consequence, trees in different bootstrap samples have similar structure (with almost always the same variables near the root) and the variance in the predictions cannot be reduced optimally. This issue is known as tree correlation.

Random forests optimally reduce the variance of the predicted values by minimising the tree correlation. This is achieved in two steps:

1. Like in bagging, different trees are fitted from bootstrap samples.
2. However, when an internal node is to be created in a given tree, the search for the optimal variable in that node is limited to only a random subset of the explanatory variables. By default, the number of variables in these subsets is one-third of the total number of variables, although this proportion is considered a tuning parameter for the model.

10.5.1 Basic implementation

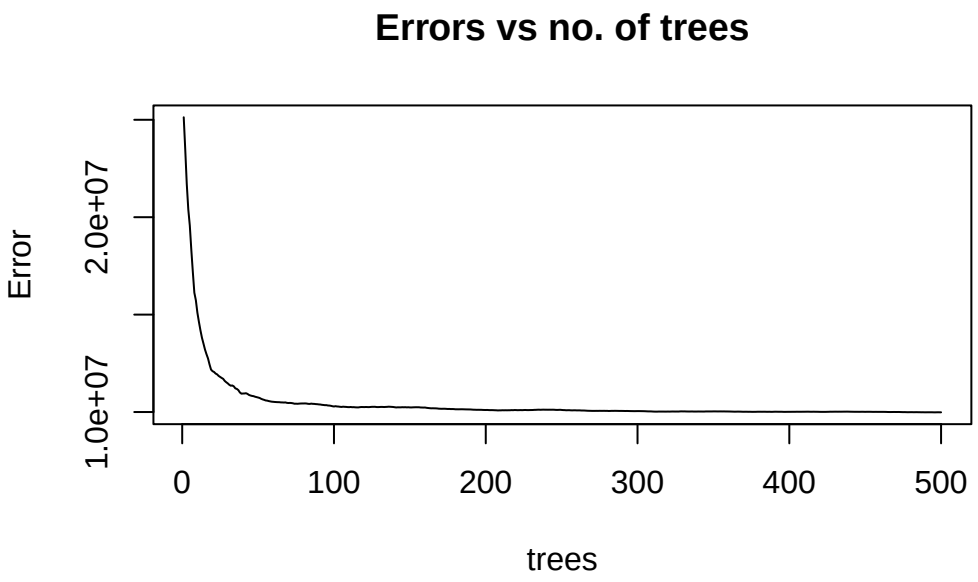
Several R implementations for random forest fitting exist, however, the most well known is provided by the `randomForest` library. By default it performs 500 trees (i.e. 500 bootstrap samples) and randomly selects one-third of the explanatory variables for each split, although these parameters can be manually tuned. The random forest model can be trained by executing just a line of code:

```
# set seed for reproducibility
set.seed(123)
# train model
fit_rf <- randomForest(formula= INCOME ~., data = df_train)
# print summary of fit
fit_rf
```

```
Call:
 randomForest(formula = INCOME ~ ., data = df_train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 16

          Mean of squared residuals: 9984866
                    % Var explained: 79.93
```

As we can see from above, the mean of squared residuals, which is the same as the MSE, is 9903736 for 500 trees and therefore, RMSE = 3147. This metric is computed by averaging residuals from the OOB samples. To illustrate how the MSE varies as more bootstrap samples are added to the model, we can plot the fitted model:

```
plot(fit_rf, main = "Errors vs no. of trees")
```



We see that the MSE becomes stable with approximately 100 trees, but it continues to decrease slowly. To find the number of trees that lead to the minimum error, we can run the following line:

```
which.min(fit_rf$mse)
```

```
[1] 496
```

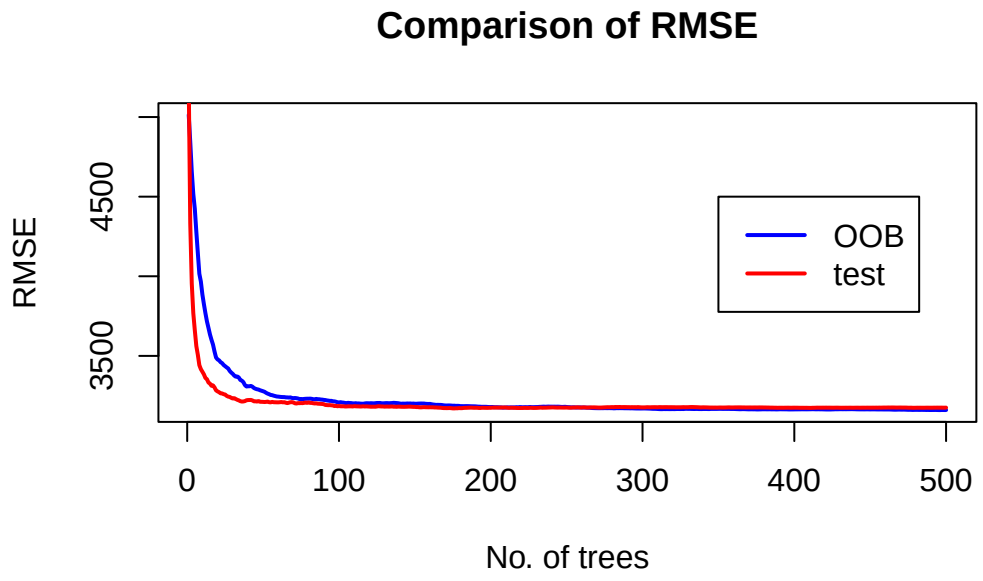
By computing the RMSE, we can compare the performance of this model with performance of the models in the previous sections:

```
sqrt(fit_rf$mse[which.min(fit_rf$mse)])
```

```
[1] 3159.537
```

This is a much lower value than what we obtained with just a single tree and even after applying bagging! Remember, this RMSE is based on the OOB samples, but we could also obtain it from the test set. `randomForest()` allows us to easily compare the RMSE obtained from OOB data and from the test set.

```r
# format test set for comparison of errors with randomForest
x_test <- df_test[setdiff(names(df_test), "INCOME")]
y_test <- df_test$INCOME

# set seed for reproducibility
set.seed(123)
# include test data in training
rf_oob_compare <- randomForest(formula = INCOME ~ ., data  = df_train, xtest = x_test, yte

# extract OOB & test errors
oob_rmse <- sqrt(rf_oob_compare$mse)
test_rmse <- sqrt(rf_oob_compare$test$mse)

# plot error vs no. of trees
x=1:rf_oob_compare$ntree
plot(x, oob_rmse, type="l", col='blue', lwd=2, xlab = "No. of trees", ylab = "RMSE", main
lines(x, test_rmse, type="l", col='red', lwd=2)
legend(350, 4500, c("OOB","test"), lwd=c(2,2), col=c("blue","red"))
```



10.5.2 Tuning

You may have noticed that the number of trees is not the only parameter we can tune in a random forest model. Below we list the model parameters, a.k.a. hyperparameters that can be tuned to improve the performance of the random tree models:

- `num.trees` is the number of trees. It should be large enough to make sure the MSE (or the RMSE) stabilises, but not too large that it creates unnecessary work.
- `mtry` is the number of variables that are randomly sampled at each split. The default is one-third of the number of variables in the original data set. If `mtry` was equal to the total number of variables, the random forest model would be equivalent to bagging. Similarly, if `mtry` was equal to 1, it would mean that only one variable is chosen, but then the results can become too biased. To find the optimal value of `mtry`, it is common to attempt 5 values evenly spread between 2 and the total number of variables in the original data set.
- `sample.fraction` controls the number of data points in each bootstrap sample, i.e. the number of samples chosen to create each tree. By default, it is 63.25% (about two-thirds) of the training set since on average, this guarantees unique data points in a sample. If the sample size is smaller, it could reduce the training time but it could also introduce some bias in the model. If the sample size is larger, it could lead to overfitting. When tuning the model, this parameter is frequently kept between 60 and 80% of the total size of the training set.
- `min.node.size` is the minimal node size to split at. Default is 5 for regression.
- `max.depth` is the maximum depth of the trees.

In order to find the combination of hyperparameters that leads to the best performing model, we need to try them all and select the one with the lowest MSE or RMSE. This is usually a computationally heavy task, so as the models and the training data become larger, the process of tuning can become very slow. The library `ranger` provides a C++ implementation of the random forest algorithm and allows to perform hyperparameter search faster than `randomForest`.

As mentioned, to find the best performing model, we need to find the right combination of hyperparameters. So the first step in the tuning process is to generate a "grid" of possible combinations of hyperparameters. If we only wanted to tune `ntree` (as we did in the previous subsection when we found that the number of trees leading to the lowest value of MSE is 495), the grid would be simply a list of possible `ntree` values. To illustrate more complex tuning, here we generate a grid that considers `mtry`, `sample.fraction` and `min.node.size`. The grid is created as follows:

```r
# Considering that there are 48 explanatory variables in the original dataset, we will try
hyper_grid <- expand.grid(mtry = seq(10, 20, by=2),
                          sample.fraction = c(0.60, 0.65, 0.70, 0.75, 0.80),
                          min.node.size = seq(3, 9, by=2))

# total number of hyperparameter combinations
nrow(hyper_grid)
```

```
[1] 120
```

Next, we can loop through the grid and generate, for each hyperparameter combination, random forest models based on 500 trees. For each random forest model, we will add the OOB RMSE error to the grid so we can find what hyperparameter combination minimises this error. Note that we set the value of seed for code reproducibility purposes.

```r
for(i in 1:nrow(hyper_grid)) {

  # train model
  fit_rf_tuning <- ranger(formula = INCOME ~ .,
    data = df_train,
    num.trees = 500,
    mtry = hyper_grid$mtry[i],
    sample.fraction = hyper_grid$sample.fraction[i],
    min.node.size = hyper_grid$min.node.size[i],
    seed = 123)

  # add OOB error to grid
  hyper_grid$OOB_RMSE[i] <- sqrt(fit_rf_tuning$prediction.error)
}
```

From the fitted models, the one that produces the minimum OOB RMSE and hence, the best-performing one, is given by the combination of parameters printed below:

```r
hyper_grid[which.min(hyper_grid$OOB_RMSE),]
```

```
   mtry sample.fraction min.node.size OOB_RMSE
26   12             0.8             3 3158.055
```

`mtry`= 14, `sample.fraction` =0.8 and `min.node.size`=3. The OOB RMSE is 3141.089, slightly lower than the error we obtained with the default model for random forest with no tuning, yay!

10.6 Questions

For this set of questions, you will use a data set very similar to the one used in the examples above. However, instead of focusing on predicting the net annual household income, you will focus in the median house price paid in each MSOA in 2021. The raw data for median house price can be downloaded here, but we have created a clean dataset for you. You can load the relevant data set by running the code below:

```
# Load the data
df_housing <- read.csv("./data/machine-learning/census2021-msoa-houseprice.csv")
# Data cleaning, remove the X field
df_housing$X <- NULL
# Data cleaning, the fields "date", "geography" and "geography.code" are not needed
df_housing <- subset(df_housing, select = -c(date, geography, geography.code))
# More data cleaning, remove comma from income and turn it into a numeric value
df_housing$HOUSEPRICE <- as.numeric(gsub(",", "", df_housing$HOUSEPRICE))
```

For the following questions, you will use the whole dataset for training and evaluate the performance of the ML models via the OOB error.

1. Train a model of decision trees with bagging using `caret`. Use 10-fold cross-validation and the default number of trees in the training process. Set the seed to 123 so that your results are reproducible. Report the OOB RMSE and the first three most important variables for the decision process in the fitted model using the function `varImp()`. Comment on why you think these three demographic variables are relatively important to determine the median house price for the MSOAs in England & Wales. What is special about these demographic characteristics? Do not include any plots.
2. Train a random forest model to predict mean house price at the MSOA level using `randomForest` with the default settings. Without including any plots, report the number of trees that produces the minimum OOB MSE. What would be the associated minimum RMSE? Like before, set the seed to 123 to ensure your results are reproducible. Do you observe any improvements in model performance with respect to the model you fitted in question 1?
3. Use `ranger` to tune the random forest model. Perform the hyperparameter search through a grid that considers only the number of trees and the number of variables to be sampled at each split. For the number of trees, try values from 490 to 500, separated by 1 unit. For the number of variables to be sampled at each split, try values from 10 to 20, also separated by 1 unit. Set the seed to 123. Without including any plots, report the combination of parameter values that leads to the model with the lowest OOB RMSE. Report the value of the OOB RMSE. Do you observe any further improvements? In the context of predicting mean house price, do you think this RMSE is acceptable? Justify your answer.

4. Using your own words, explain what are the advantages of using random forests to predict median house prices instead of multilinear regression. You should include references to the ML literature to support your arguments.

11 Data sets

```
library(sf)
library(dplyr)
```

11.1 Greater Machester land use data

Availability

The dataset is stored on a gpkg file that can be found, within the structure of this project, under:

```
st_LSOA <- st_read("./data/geodemographics/manchester_land_cover_2011.gpkg")
```

```
Reading layer `manchester_land_cover_2011' from data source
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202324/r4ps/data/geodemographi
  using driver `GPKG'
Simple feature collection with 1673 features and 44 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 351662.3 ymin: 381166 xmax: 406087.2 ymax: 421037.7
Projected CRS: OSGB36 / British National Grid
```

Variables

The variables included in this dataset follow the land use classification of the CORINE Land Cover dataset.

Source & Pre-processing

The data was sourced from What do 'left behind' areas look like over time? and cleaned on Python.

11.2 British administrative boundaries (LSOAs and LAs)

Availability

The dataset for the boundaries of the lower-layer super-output areas (LSOAs) within London is stored as a shapefile that can be found under:

```
st_LSOA <- st_read("data/geodemographics-old/LSOA_2011_London_gen_MHW/LSOA_2011_London_gen
```

```
Reading layer `LSOA_2011_London_gen_MHW' from data source
  `/Users/franciscorowe/Dropbox/Francisco/uol/teaching/envs418/202324/r4ps/data/geodemographi
  using driver `ESRI Shapefile'
Simple feature collection with 4835 features and 14 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 503574.2 ymin: 155850.8 xmax: 561956.7 ymax: 200933.6
Projected CRS: OSGB36 / British National Grid
```

The dataset for the boundaries of the local authority distrits (LADs) for the UK is stored as a shapefile that can be found under:

```
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/envs418/202324/r4ps/data/networks/Loca
  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
```

Variables

For each of the 4,835 LSOAs, the following characteristics are available:

```
names(st_LSOA)
```

```
 [1] "LSOA11CD"  "LSOA11NM"  "MSOA11CD"  "MSOA11NM"  "LAD11CD"   "LAD11NM"
 [7] "RGN11CD"   "RGN11NM"   "USUALRES"  "HHOLDRES"  "COMESTRES" "POPDEN"
[13] "HHOLDS"    "AVHHOLDSZ" "geometry"
```

where:

- `LSOA11CD`: Lower-Layer Super-Output Area code
- `LSOA11NM`: Lower-Layer Super-Output Area code
- `MSOA11CD`: Medium-Layer Super-Output Area code
- `MSOA11NM`: Medium-Layer Super-Output Area code
- `LAD11CD`: Local Authority District code
- `LAD11NM`: Local Authority District name
- `RGN11CD`: Region code
- `RGN11NM`: Region name
- `USUALRES`: Usual residents
- `HHOLDRES`: Household residents
- `COMESTRES`: Communal Establishment residents
- `POPDEN`: Population density
- `HHOLDS`: Number of households
- `AVHHOLDSZ`: Average household size
- `geometry`: Polygon of LSOA

For each of the 374 LADs, the following characteristics are available:

```
names(LA_UK)
```

```
 [1] "OBJECTID"   "LAD22CD"    "LAD22NM"    "BNG_E"      "BNG_N"
 [6] "LONG"       "LAT"        "GlobalID"   "SHAPE_Leng" "SHAPE_Area"
[11] "geometry"
```

where:

- `OBJECTID`: object identifier
- `LAD22CD`: Local Authority District code
- `LAD22NM`: Local Authority District name
- `BNG_E`: Location Easting
- `BNG_N`: Location Northing
- `LONG`: Location Longitude
- `LAT`: Location Latitude
- `GlobalID`: Global Identifier
- `SHAPE_Leng`: Boundary length
- `SHAPE_Area`: Area within boundary
- `geometry`: Polygon of LAD

Projection

The shapes of each LSOA are stored as polygons an expressed in the OSGB36 projection:

```
st_crs(st_LSOA)
```

```
Coordinate Reference System:
  User input: OSGB36 / British National Grid
  wkt:
PROJCRS["OSGB36 / British National Grid",
    BASEGEOGCRS["OSGB36",
        DATUM["Ordnance Survey of Great Britain 1936",
            ELLIPSOID["Airy 1830",6377563.396,299.3249646,
                LENGTHUNIT["metre",1]],
            ID["EPSG",6277]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["Degree",0.0174532925199433]]],
    CONVERSION["unnamed",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",49,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",-2,
            ANGLEUNIT["Degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.999601272,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",400000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",-100000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1,
                ID["EPSG",9001]]],
        AXIS["(N)",north,
            ORDER[2],
```

```
            LENGTHUNIT["metre",1,
                ID["EPSG",9001]]]]
```

Similarly, the shapes of each LAD are stored as polygons an expressed in the OSGB36 projection:

```
st_crs(LA_UK)
```

```
Coordinate Reference System:
  User input: OSGB36 / British National Grid
  wkt:
PROJCRS["OSGB36 / British National Grid",
    BASEGEOGCRS["OSGB36",
        DATUM["Ordnance Survey of Great Britain 1936",
            ELLIPSOID["Airy 1830",6377563.396,299.3249646,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4277]],
    CONVERSION["British National Grid",
        METHOD["Transverse Mercator",
            ID["EPSG",9807]],
        PARAMETER["Latitude of natural origin",49,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8801]],
        PARAMETER["Longitude of natural origin",-2,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8802]],
        PARAMETER["Scale factor at natural origin",0.9996012717,
            SCALEUNIT["unity",1],
            ID["EPSG",8805]],
        PARAMETER["False easting",400000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8806]],
        PARAMETER["False northing",-100000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8807]]],
    CS[Cartesian,2],
        AXIS["(E)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["(N)",north,
```

```
        ORDER[2],
        LENGTHUNIT["metre",1]],
USAGE[
    SCOPE["Engineering survey, topographic mapping."],
    AREA["United Kingdom (UK) - offshore to boundary of UKCS within 49°45'N to 61°N and 
    BBOX[49.75,-9,61.01,2.01]],
ID["EPSG",27700]]
```

Source & Pre-processing

The boundaries for the LSOAs within London can be found directly from the London Datastore website.

The boundaries for the LADs for the UK can be found on the ONS Open Geography Portal website. To filter for the London LADs, i.e. the London boroughs, we run the following line of code:

```
LND_boroughs <- LA_UK %>% filter(grepl('E09', LAD22CD))
```

11.3 Worldpop population count data for Ukraine

11.4 Census population count data for UK

11.5 Ukraine's administrative boundaries

11.6 Internal migration flows between US metropolitan areas and between London boroughs

Availability

The dataset for the migration flows between US metropolitan areas can be found as a csv file under:

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

The dataset for the migration flows between London boroughs can be found as a csv file under:

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

Variables

For each of the 52,930 movements recorded on the dataset for the migration flows between US metropolitan areas, the following fields are available:

```
names(df_metro)
```

```
 [1] "MSA_Current_Code"
 [2] "MSA_Current_Name"
 [3] "MSA_Current_State"
 [4] "MSA_Current_Population_1_Year_and_Over_Estimate"
 [5] "MSA_Current_Population_1_Year_and_Over_MOE"
 [6] "MSA_Previous_Code"
 [7] "MSA_Previous_Name"
 [8] "MSA_Previous_State"
 [9] "MSA_Previous_Population_1_Year_and_Over_Estimate"
[10] "MSA_Previous_Population_1_Year_and_Over_MOE"
[11] "Movers_Metro_to_Metro_Flow_Estimate"
[12] "Movers_Metro_to_Metro_Flow_MOE"
```

All the fields that start with `MSA_Current_` or `MSA_Previous_` refer to the characteristics of the origin and destination metropolican areas. The relevant fields for the analysis in this book are:

- `Movers_Metro_to_Metro_Flow_Estimate`: Estimate of number of people moving between origin and destination
- `Movers_Metro_to_Metro_Flow_MOE`: Margin of error for the above estimate

More details on the methodology to obtain the estimates and the margin of error for each population movement can be found on the US Census Bureau website.

For each of the 1,053 movements recorded on the dataset for the migration flows between London boroughs, the following fields are available:

```
names(df_borough)
```

```
[1] "OutLA" "InLA"  "Moves"
```

where:

- `OutLA` is the code corresponding to the origin borough
- `InLA` is the code corresponding to the destination borough
- `Moves` is the number of internal migration moves within each flow. Note that the numbers are not integers. This is because of the various scaling processes used to produce the dataset, which are described in more detail in the latest methodology document, which can be found here.

Source & pre-processing

The dataset for the migration flows between US metropolitan areas can be downloaded from the US Census Bureau website. The data was cleaned on Microsoft Excel.

The dataset for the migration flows between London boroughs can be downloaded from the ONS website. The data was cleaned on Microsoft Excel.

11.7 Twitter data on public opinion originated in the US and in the UK

11.8 Reddit data

11.9 Google mobility data for Italy and the UK

11.10 COVID-19 cases data for London and Rome

11.11 Census MSOA data for England and Wales

Availability

The dataset for the demographic census data of each MSOA in England and Wales can be loaded as a csv file from:

```
df_MSOA <- read.csv("./data/machine-learning/census2021-msoa-income.csv")
```

A very similar dataset for the demographic census data of each MSOA in England and Wales which also contains data on the median house price can be loaded as a csv file from:

```
df_housing <- read.csv("./data/machine-learning/census2021-msoa-houseprice.csv")
```

Variables

For each of the 7,080 MSOAs recorded in England and Wales, the following fields are available:

```
names(df_MSOA)
```

```
 [1] "X"                 "date"              "geography"         "geography.code"
 [5] "inHH"              "inCE"              "SING"              "MARRIED"
 [9] "SEP"               "DIV"               "WIDOW"             "UK"
[13] "EU"                "AFR"               "AS"                "AM"
[17] "OC"                "BO"                "DENSITY"           "Y14orUNDER"
[21] "Y15to19"           "Y20to24"           "Y25to29"           "Y30to34"
[25] "Y35to49"           "Y40to44"           "Y45to49"           "Y50to54"
[29] "Y55to59"           "Y60to64"           "Y65orOVER"         "F"
[33] "M"                 "HH1"               "HH2"               "HH3"
[37] "HH4"               "HH5"               "HH6"               "ADD1YagoSAME"
[41] "ADD1YagoSTUDENT"   "ADD1YagoUK"        "ADD1YagoNONUK"     "NHH"
[45] "OWN"               "MORTGAGE"          "SHAREDOWN"         "RENTfromCOUNCIL"
[49] "RENTotherSOCIAL"   "RENTprivate"       "RENTprivateOTHER"  "RENTfree"
[53] "INCOME"
```

For a description of the variables in the columns of df_MSOA, we can load a dictionary for these variables:

```
df_dictionary <- read.csv("./data/machine-learning/Dictionary.csv")
head(df_dictionary)
```

```
                                    Dictionary       X
1                                                    
2                                          Name     Key
3                 Lives in household (% persons)    inHH
4    Lives in communal establishment (% persons)    inCE
5 Never married or civil partnership (% persons)    SING
6    Married or in civil partnership (% persons) MARRIED
```

Source & pre-processing

Data on the the census characteristics for different MSOAs can be downloaded from the Nomis website. Data on the average net household income can be obtained from the ONS website.

Data on the median houseprice for different MSOAs can be downloaded from the ONS website.

All the data has been pre-processed on Microsoft Excel.

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.

Bar, Michael, Moshe Hazan, Oksana Leukhina, David Weiss, and Hosny Zoabi. 2018. "Why Did Rich Families Increase Their Fertility? Inequality and Marketization of Child Care." *Journal of Economic Growth* 23: 427–63.

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

Blei, David M, Andrew Y Ng, and Michael I Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* 3 (Jan): 993–1022.

Boehmke, Brad. 2019. "Hands-on Machine Learning with r. Chapter 9: Decision Trees." https://bradleyboehmke.github.io/HOML/DT.html.

Breiman, L. 1984. *Classification and Regression Trees (1st Ed.)*. Routledge.

Brodeur, Abel, Andrew E Clark, Sarah Fleche, and Nattavudh Powdthavee. 2021. "COVID-19, Lockdowns and Well-Being: Evidence from Google Trends." *Journal of Public Economics* 193: 104346.

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.

Cinelli, Matteo, Walter Quattrociocchi, Alessandro Galeazzi, Carlo Michele Valensise, Emanuele Brugnoli, Ana Lucia Schmidt, Paola Zola, Fabiana Zollo, and Antonio Scala. 2020. "The COVID-19 Social Media Infodemic." *Scientific Reports* 10 (1): 1–10.

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.

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.

Goodman-Bacon, Andrew, and Jan Marcus. 2020. "Using Difference-in-Differences to Identify Causal Effects of COVID-19 Policies."

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

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

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.

Marshall, Emily A. 2013. "Defining Population Problems: Using Topic Models for Cross-National Comparison of Disciplinary Development." *Poetics* 41 (6): 701–24.

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.

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, Katariina, 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.

Schleicher, Andreas. 2020. "The Impact of COVID-19 on Education: Insights from" Education at a Glance 2020"." *OECD Publishing*.

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.

Sielge, Jullia, and David Robinson. 2022. *Welcome to Text Mining with r*. O'Reilly. https://www.tidytextmining.com.

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.

Ugolini, Francesca, Luciano Massetti, Pedro Calaza-Martínez, Paloma Cariñanos, Cynnamon Dobbs, Silvija Krajter Ostoić, Ana Marija Marin, et al. 2020. "Effects of the COVID-19 Pandemic on the Use and Perceptions of Urban Green Space: An International Exploratory Study." *Urban Forestry & Urban Greening* 56: 126888.

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.

Zhou, Muzhi, and Man-Yee Kan. 2021. "The Varying Impacts of COVID-19 and Its Related Measures in the UK: A Year in Review." *PLoS One* 16 (9): e0257286.