Contents

Welcome 7  
Contact ....................................................... 7

1 Overview 9  
1.1 Aims ....................................................... 9  
1.2 Learning Outcomes ...................................... 10  
1.3 Feedback .................................................. 10  
1.4 Computational Environment .................................. 10  
1.5 Assessment ................................................. 11

2 Spatial Data 15  
2.1 Spatial Data types ........................................... 15  
2.2 Hierarchical Structure of Data .................................. 17  
2.3 Key Challenges ................................................ 18

3 Data Wrangling 21  
3.1 Dependencies ............................................... 21  
3.2 Introducing R ............................................... 22  
3.3 Setting the working directory .............................. 24  
3.4 R Scripts and Notebooks ...................................... 24  
3.5 Getting Help .................................................. 27  
3.6 Variables and objects ....................................... 28  
3.7 Data Frames ............................................... 31  
3.8 Read Data ..................................................... 33  
3.9 Manipulation Data ............................................ 35  
3.10 Using Spatial Data Frames .................................... 37  
3.11 Useful Functions ............................................. 43

4 Points 45  
4.1 Dependencies ............................................... 45  
4.2 Data ......................................................... 46  
4.3 Binning ....................................................... 48  
4.4 KDE ......................................................... 50  
4.5 Spatial Interpolation .......................................... 53
# CONTENTS

4.6 Questions .................................................. 60

5 Flows .................................................. 63
  5.1 Dependencies ........................................... 63
  5.2 Data ................................................... 65
  5.3 “Seeing” flows ........................................... 66
  5.4 Modelling flows ......................................... 72
  5.5 Predicting flows ......................................... 88
  5.6 Questions ................................................ 90

6 Spatial Econometrics ....................................... 93
  6.1 Dependencies ........................................... 93
  6.2 Data ................................................... 94
  6.3 Non-spatial regression, a refresh ......................... 95
  6.4 Spatial regression: a (very) first dip ..................... 97
  6.5 Spatial heterogeneity .................................... 98
  6.6 Spatial dependence ..................................... 111
  6.7 Predicting house prices ................................ 115
  6.8 Questions ................................................ 116

7 Multilevel Modelling - Part 1 ............................. 119
  7.1 Dependencies ........................................... 119
  7.2 Data ................................................... 120
  7.3 Modelling ............................................... 124
  7.4 Multilevel Modelling: Random Intercept Model ........... 127
  7.5 Questions ................................................ 142

8 Multilevel Modelling - Part 2 ............................. 147
  8.1 Dependencies ........................................... 147
  8.2 Data ................................................... 148
  8.3 Conceptual Overview ................................... 148
  8.4 Estimating Varying Intercept and Slopes Models .......... 151
  8.5 Interpreting Correlations Between Group-level Intercepts and Slopes .......................................................... 157
  8.6 Model building .......................................... 158
  8.7 Questions ................................................ 159

9 Geographically Weighted Regression ....................... 161
  9.1 Dependencies ........................................... 161
  9.2 Data ................................................... 162
  9.3 Recap: Spatial Effects ................................... 163
  9.4 Exploratory Analysis ..................................... 163
  9.5 Global Regression ....................................... 166
  9.6 Fitting a Geographically Weighted Regression ............. 171
  9.7 Questions ................................................ 183

10 Spatio-Temporal Analysis ................................ 185
Welcome

This is the website for “Spatial Modeling for Data Scientists”. This is a course taught by Dr. Francisco Rowe and Dr. Dani Arribas-Bel in the Second Semester of 2020/21 at the University of Liverpool, United Kingdom. You will learn how to analyse and model different types of spatial data as well as gaining an understanding of the various challenges arising from manipulating such data.

The website is licensed under the Attribution-NonCommercial-NoDerivatives 4.0 International License. A compilation of this web course is hosted as a GitHub repository that you can access:

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

Contact

Francisco Rowe - F.Rowe-Gonzalez [at] liverpool.ac.uk
Senior Lecturer in Quantitative Human Geography
Office 507, Roxby Building,
University of Liverpool - 74 Bedford St S,
Liverpool, L69 7ZT,
United Kingdom.

Dani Arribas-Bel - D.Arribas-Bel [at] liverpool.ac.uk
Senior Lecturer in Geographic Data Science
Office 508, Roxby Building,
University of Liverpool - 74 Bedford St S,
Liverpool, L69 7ZT,
United Kingdom.
Chapter 1

Overview

Access to all materials, including lecture notes, computational notebooks and datasets, is centralised through the use of the course website available in the following url:

https://gdsl-ul.github.io/san/

The module handbook, including the assessment description, criteria and module programme, and videos for each teaching week can be accessed via the module Canvas site:

ENS453 Spatial Modelling for Data Scientists

1.1 Aims

This module aims to provide students with a range of techniques for analysing and modelling spatial data:

- build upon the more general research training delivered via companion modules on *Data Collection and Data Analysis*, both of which have an aspatial focus;
- highlight a number of key social issues that have a spatial dimension;
- explain the specific challenges faced when attempting to analyse spatial data;
- introduce a range of analytical techniques and approaches suitable for the analysis of spatial data; and,
- enhance practical skills in using R software packages to implement a wide range of spatial analytical tools.
1.2 Learning Outcomes

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

- identify some key sources of spatial data and resources of spatial analysis and modelling tools;
- explain the advantages of taking spatial structure into account when analysing spatial data;
- apply a range of computer-based techniques for the analysis of spatial data, including mapping, correlation, kernel density estimation, regression, multi-level models, geographically-weighted regression, spatial interaction models and spatial econometrics;
- apply appropriate analytical strategies to tackle the key methodological challenges facing spatial analysis – spatial autocorrelation, heterogeneity, and ecological fallacy; and,
- select appropriate analytical tools for analysing specific spatial data sets to address emerging social issues facing the society.

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 lecture, 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 maintained by the module lead. 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.

1.4 Computational Environment

To reproduce the code in the book, you need the most recent version of R and packages. These can be installed following the instructions provided in our R installation guide.

1.4.1 Dependency list

The list of libraries used in this book is provided below. If you have followed the instructions provided in our R installation guide and will be using Docker,
you can relax and these libraries have already been installed for you.

If you have natively installed R and RStudio, you need to ensure you have installed the list of libraries used in this book following the steps provided here.

- arm
- car
- corrplot
- FRK
- gghighlight
- ggplot2
- ggmap
- GISTools
- gridExtra
- gstat
- jtools
- kableExtra
- knitr
- lme4
- lmtest
- lubridate
- MASS
- merTools
- plyr
- RColorBrewer
- rgdal
- sf
- sjPlot
- sp
- spgwr
- spatialreg
- spacetime
- stargazer
- tidyverse
- tmap
- viridis

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%) refer to the set of questions at the end of Chapters 4, 5 and 6. 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%) refer to the set of questions at the end of Chapters 7, 8, 9 and 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 R Notebook format and then converted into a self-contained HTML file that will then be submitted via Turnitin. The notebook 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 the R Markdown website from RStudio:
- A Guide
- R Markdown: The Definitive Guide by Xie et al. (2018)
- R Markdown Reference Guide

Two R Notebook templates will be available via the module Canvas site.

Submission is electronic only via Turnitin on Canvas.

1.5.2 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.
1.5. ASSESSMENT

- **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.).
Chapter 2

Spatial Data

This Chapter seeks to present and describe distinctive attributes of spatial data, and discuss some of the main challenges in analysing and modelling these data. Spatial data is a term used to describe any data associating a given variable attribute to a specific location on the Earth’s surface.

2.1 Spatial Data types

Different classifications of spatial data types exist. Knowing the structure of the data at hand is important as specific analytical methods would be more appropriate for particular data types. We will use a particular classification involving four data types: lattice/areal data, point data, flow data and trajectory data (Fig. 1). This is not a exhaustive list but it is helpful to motivate the analytical and modelling methods that we cover in this book.

Lattice/Areal Data. These data correspond to records of attribute values (such as population counts) for a fixed geographical area. They may comprise regular shapes (such as grids or pixels) or irregular shapes (such as states, counties or travel-to-work areas). Raster data are a common source of regular lattice/areal area, while censuses are probably the most common form of irregular lattice/areal area. Point data within an area can be aggregated to produce lattice/areal data.

Point Data. These data refer to records of the geographic location of an discrete event, or the number of occurrences of geographical process at a given location. As displayed in Fig. 1, examples include the geographic location of bus stops in a city, or the number of boarding passengers at each bus stop.

Flow Data. These data refer to records of measurements for a pair of geographic point locations. or pair of areas. These data capture the linkage or spatial


Figure 2.1: Fig. 1. Data Types. Area / Lattice data source: Önnerfors et al. (2019). Point data source: Tao et al. (2018). Flow data source: Rowe and Patias (2020). Trajectory data source: Kwan and Lee (2004).
interaction between two locations. Migration flows between a place of origin and a place of destination is an example of this type of data.

*Trajectory Data.* These data record geographic locations of moving objects at various points in time. A trajectory is composed of a single string of data recording the geographic location of an object at various points in time and each record in the string contains a time stamp. These data are complex and can be classified into explicit trajectory data and implicit trajectory data. The former refer to well-structured data and record positions of objects continuously and intensively at uniform time intervals, such as GPS data. The latter is less structured and record data in relatively time point intervals, including sensor-based, network-based and signal-based data (Kong et al. (2018)).

In this course, we cover analytical and modelling approaches for point, lattice/areal and flow data. While we do not explicitly analyse trajectory data, various of the analytical approaches described in this book can be extended to incorporate time, and can be applied to model these types of data. In Chapter 10, we describe approaches to analyse and model spatio-temporal data. These same methods can be applied to trajectory data.

### 2.2 Hierarchical Structure of Data

The hierarchical organisation is a key feature of spatial data. Smaller geographical units are organised within larger geographical units. You can find the hierarchical representation of UK Statistical Geographies on the Office for National Statistics website. In the bottom part of the output below, we can observe a spatial data frame for Liverpool displaying the hierarchical structure of census data (from the smallest to the largest): Output Areas (OAs), Lower Super Output Areas (LSOAs), Middle Super Output Areas (MSOAs) and Local Authority Districts (LADs). This hierarchical structure entails that units in smaller geographies are nested within units in larger geographies, and that smaller units can be aggregated to produce large units.

```r
## Simple feature collection with 6 features and 4 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox: xmin: 335071.6 ymin: 389876.7 xmax: 339426.9 ymax: 394479
## projected CRS: Transverse_Mercator
## OA_CD    LSOA_CD    MSOA_CD    LAD_CD    geometry
## 1 E00176737 E01033761 E02006932 E08000012 MULTIPOLYGON (((335106.3 389448...)
## 2 E00033515 E01006614 E02001358 E08000012 MULTIPOLYGON (((335810.5 390053...)
## 3 E00033141 E01006546 E02001365 E08000012 MULTIPOLYGON (((336738 393106...)
## 4 E00176757 E01006646 E02001369 E08000012 MULTIPOLYGON (((335914.5 391451...)
## 5 E00034050 E01006712 E02001375 E08000012 MULTIPOLYGON (((339325 391451...)
## 6 E00034280 E01006761 E02001366 E08000012 MULTIPOLYGON (((338198.1 390683...)
```

Next we quickly go through the components of the output above. The first line
indicates the type of feature and the number of rows (features) and columns (fields) in the data frame, except for the geometry. The second and third lines identify the type of geometry and dimension. The fourth line \texttt{bbox} stands for bounding box and display the min and max coordinates containing the Liverpool area in the data frame. The fifth line \texttt{projected CRS} indicates the coordinate reference system projection. If you would like to learn more about the various components of spatial data frames, please see the \texttt{R sf} package vignette on Simple Features.

\section*{2.3 Key Challenges}

Major challenges exist when working with spatial data. Below we explore some of the key longstanding problems data scientists often face when working with geographical data.

\subsection*{2.3.1 Modifiable Area Unit Problem (MAUP)}

The Modifiable Area Unit Problem (MAUP) represents a challenge that has troubled geographers for decades (Openshaw, 1981). Two aspects of the MAUP are normally recognised in empirical analysis relating to \textit{scale} and \textit{zonation}. Fig. 2 illustrates these issues

- \textit{Scale} refers to the idea that a geographical area can be divided into geographies with differing numbers of spatial units.

- \textit{Zonation} refers to the idea that a geographical area can be divided into the same number of units in a variety of ways.

The MAUP is a critical issue as it can impact our analysis and thus any conclusions we can infer from our results (e.g. Fotheringham and Wong, 1991). There is no agreed systematic approach on how to handle the effects of the MAUP. Some have suggested to perform analyses based on different existing geographical scales, and assess the consistency of the results and identify potential sources of change. The issue with such approach is that results from analysis at different scales are likely to differ because distinct dimensions of a geographic process may be captured at different scales. For example, in migration studies, smaller geographies may be more suitable to capture residential mobility over short distances, while large geographies may be more suitable to capture long-distance migration. And it is well documented that these types of moves are driven by different factors. While residential mobility tends to be driven by housing related reasons, long-distance migration is more closely related to employment-related motives (Niedomysl, 2011).

An alternative approach is to use the smallest geographical system available and create random aggregations at various geographical scales, to directly quantify the extent of scale and zonation. This approach has shown promising results in applications to study internal migration flows (Stillwell et al., 2018). Another
2.3. KEY CHALLENGES

2.3.2 Ecological Fallacy

Ecological fallacy is an error in the interpretation of statistical data based on aggregate information. Specifically it refers to inferences made about the nature of specific individuals based solely on statistics aggregated for a given group. It is about thinking that relationships observed for groups necessarily hold for individuals. A key example is Robinson (1950) who illustrates this problem exploring the difference between ecological correlations and individual correlations. He looked at the relationship between country of birth and literacy. Robinson (1950) used the percent of foreign-born population and percent of literate population for the 48 states in the United States in 1930. The ecological correlation based on these data was 0.53. This suggests a positive association between foreign birth and literacy, and could be interpreted as foreign born individuals being more likely to be literate than native-born individuals. Yet, the correlation based on individual data was negative -0.11 which indicates the opposite. The main point emerging from this example is to carefully interpret analysis.
based on spatial data and avoid making inferences about individuals from these data.

2.3.3 Spatial Dependence

Spatial dependence refers to the spatial relationship of a variable’s values for a pair of locations at a certain distance apart, so that these values are more similar (or less similar) than expected for randomly associated pairs of observations (Anselin, 1988). For example, we could think of observed patterns of ethnic segregation in an area are a result of spillover effects of pre-existing patterns of ethnic segregation in neighbouring areas. Chapter 6 will illustrate approach to explicitly incorporate spatial dependence in regression analysis.

2.3.4 Spatial Heterogeneity

Spatial heterogeneity refers to the uneven distribution of a variable’s values across space. Concentration of deprivation or unemployment across an area are good examples of spatial heterogeneity. We illustrate various ways to visualise, explore and measure the spatial distribution of data in multiple chapters. We also discuss on potential modelling approaches to capture spatial heterogeneity in Chapters 6, 7 and 10.

2.3.5 Spatial nonstationarity

Spatial nonstationarity refers to variations in the relationship between an outcome variable and a set of predictor variables across space. In a modelling context, it relates to a situation in which a simple “global” model is inappropriate to explain the relationships between a set of variables. The geographical nature of the model must be modified to reflect local structural relationships within the data. For example, ethnic segregation has been positively associated with employment outcomes in some countries pointing to networks in pre-existing communities facilitating access to the local labour market. Inversely ethnic segregation has been negatively associated with employment outcomes pointing to lack of integration into the broader local community. We illustrate various modelling approaches to capture spatial nonstationarity in Chapters 8 and 9.
Chapter 3

Data Wrangling

This chapter introduces R Notebooks, basic functions and data types. These are all important concepts that we will use during the module.

If you are already familiar with R, R notebooks and data types, you may want to jump to Section Read Data and start from there. This section describes how to read and manipulate data using sf and tidyverse functions, including mutate(), %>% (known as pipe operator), select(), filter() and specific packages and functions how to manipulate spatial data.

The chapter is based on:

- Grolemund and Wickham (2019), this book illustrates key libraries, including tidyverse, and functions for data manipulation in R
- Xie et al. (2019), excellent introduction to R markdown!
- Williamson (2018), some examples from the first lecture of ENVS450 are used to explain the various types of random variables.
- Lovelace et al. (2020), a really good book on handling spatial data and historical background of the evolution of R packages for spatial data analysis.

---

1This chapter is part of Spatial Analysis Notes Introduction – R Notebooks + Basic Functions + Data Types by Francisco Rowe is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.
CHAPTER 3. DATA WRANGLING

3.1 Dependencies

This tutorial uses the libraries below. Ensure they are installed on your machine\(^2\) before loading them executing the following code chunk:

```r
# Data manipulation, transformation and visualisation
library(tidyverse)

## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1
# Nice tables
library(kableExtra)
# Simple features (a standardised way to encode vector data ie. points, lines, polygons)
library(sf)
# Spatial objects conversion
library(sp)
# Thematic maps
library(tmap)
# Colour palettes
library(RColorBrewer)
# More colour palettes
library(viridis)
```

3.2 Introducing R

R is a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques. It has gained widespread use in academia and industry. R offers a wider array of functionality than a traditional statistics package, such as SPSS and is composed of core (base) functionality, and is expandable through libraries hosted on CRAN. CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R.

Commands are sent to R using either the terminal / command line or the R Console which is installed with R on either Windows or OS X. On Linux, there is no equivalent of the console; however, third party solutions exist. On your own machine, R can be installed from here.

Normally RStudio is used to implement R coding. RStudio is an integrated development environment (IDE) for R and provides a more user-friendly front-end to R than the front-end provided with R.

To run R or RStudio, just double click on the R or RStudio icon. Throughout this module, we will be using RStudio:

\(^2\)You can install package `mypackage` by running the command `install.packages("mypackage")` on the R prompt or through the Tools --> Install Packages... menu in RStudio.
3.2. INTRODUCING R

Figure 3.1: Fig. 1. RStudio features.
If you would like to know more about the various features of RStudio, watch this video.

### 3.3 Setting the working directory

Before we start any analysis, ensure to set the path to the directory where we are working. We can easily do that with `setwd()`. Please replace in the following line the path to the folder where you have placed this file -and where the data folder lives.

```r
#setwd('..//data/sar.csv')
#setwd('.')
```

Note: It is good practice to not include spaces when naming folders and files. Use underscores or dots.

You can check your current working directory by typing:

```r
getwd()
```

## [1] "/home/jovyan/work"

### 3.4 R Scripts and Notebooks

An *R script* is a series of commands that you can execute at one time and help you save time. So you don’t repeat the same steps every time you want to execute the same process with different datasets. An R script is just a plain text file with R commands in it.

To create an R script in RStudio, you need to

- Open a new script file: *File > New File > R Script*
- Write some code on your new script window by typing eg. `mtcars`
- Run the script. Click anywhere on the line of code, then hit *Ctrl + Enter* (Windows) or *Cmd + Enter* (Mac) to run the command or select the code chunk and click *run* on the right-top corner of your script window. If do that, you should get:

```
mtcars
```

<table>
<thead>
<tr>
<th>mpg</th>
<th>cyl</th>
<th>disp</th>
<th>hp</th>
<th>drat</th>
<th>wt</th>
<th>qsec</th>
<th>vs</th>
<th>am</th>
<th>gear</th>
<th>carb</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.0</td>
<td>6</td>
<td>160.0</td>
<td>110</td>
<td>3.90</td>
<td>2.620</td>
<td>16.46</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>21.0</td>
<td>6</td>
<td>160.0</td>
<td>110</td>
<td>3.90</td>
<td>2.875</td>
<td>17.02</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>22.8</td>
<td>4</td>
<td>108.0</td>
<td>93</td>
<td>3.85</td>
<td>2.320</td>
<td>18.61</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>21.4</td>
<td>6</td>
<td>258.0</td>
<td>110</td>
<td>3.08</td>
<td>3.215</td>
<td>19.44</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>18.7</td>
<td>8</td>
<td>360.0</td>
<td>175</td>
<td>3.15</td>
<td>3.440</td>
<td>17.02</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>18.1</td>
<td>6</td>
<td>225.0</td>
<td>105</td>
<td>2.76</td>
<td>3.460</td>
<td>20.22</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>14.3</td>
<td>8</td>
<td>360.0</td>
<td>245</td>
<td>3.21</td>
<td>3.570</td>
<td>15.84</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
3.4. R SCRIPTS AND NOTEBOOKS

```
## Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2
## Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2
## Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4
## Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1 0 4 4
## Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3
## Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3
## Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3
## Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4
## Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4
## Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4
## Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1
## Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2
## Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1
## Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1
## Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2
## AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2
## Camaro Z28 13.3 8 350.0 245 3.73 3.845 15.41 0 0 3 4
## Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2
## Fiat X1-9 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1
## Porsche 914-2 26.0 8 120.3 91 4.43 2.140 16.70 0 1 5 2
## Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2
## Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4
## Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6
## Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8
## Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2
```

• Save the script: File > Save As, select your required destination folder, and enter any filename that you like, provided that it ends with the file extension .R

An R Notebook is an R Markdown document with descriptive text and code chunks that can be executed independently and interactively, with output visible immediately beneath a code chunk - see Xie et al. (2019).

To create an R Notebook, you need to:

• Open a new script file: File > New File > R Notebook

• Insert code chunks, either:

  1) use the Insert command on the editor toolbar;
  2) use the keyboard shortcut Ctrl + Alt + I or Cmd + Option + I (Mac);
  or,
  3) type the chunk delimiters ```{r} and ```

In a chunk code you can produce text output, tables, graphics and write code! You can control these outputs via chunk options which are provided inside the curly brackets eg.
CHAPTER 3. DATA WRANGLING

---
title: "My Notebook"
output: html_notebook
---

Figure 3.2: Fig. 2. YAML metadata for notebooks.

```
[r, include=TRUE, echo=TRUE, fig.margin=FALSE]
hist(mtcars$mpg)
```

Figure 3.3: Fig. 3. Code chunk example. Details on the various options: https://rmarkdown.rstudio.com/lesson-3.html
3.5. GETTING HELP

- Execute code: hit “Run Current Chunk”, Ctrl + Shift + Enter or Cmd + Shift + Enter (Mac)
- Save an R notebook: File > Save As. A notebook has a *.Rmd extension and when it is saved a *.nb.html file is automatically created. The latter is a self-contained HTML file which contains both a rendered copy of the notebook with all current chunk outputs and a copy of the *.Rmd file itself.

Rstudio also offers a Preview option on the toolbar which can be used to create pdf, html and word versions of the notebook. To do this, choose from the drop-down list menu knit to ...

3.5 Getting Help

You can use help or ? to ask for details for a specific function:

\[
\text{help(sqrt) \#or \ ?sqrt}
\]

And using example provides examples for said function:

\[
\text{example(sqrt)}
\]

##
## sqrt> require(stats) \# for spline
##
## sqrt> require(graphics)
##
## sqrt> xx <- -9:9
##
## sqrt> plot(xx, sqrt(abs(xx)), col = "red")
##
## sqrt> lines(spline(xx, sqrt(abs(xx)), n=101), col = "pink")

3.6 Variables and objects

An object is a data structure having attributes and methods. In fact, everything in R is an object!

A variable is a type of data object. Data objects also include list, vector, matrices and text.

- Creating a data object

In R a variable can be created by using the symbol <- to assign a value to a variable name. The variable name is entered on the left <- and the value on the right. Note: Data objects can be given any name, provided that they start with a letter of the alphabet, and include only letters of the alphabet, numbers and
the characters , and _ . Hence AgeGroup, Age_Group and Age.Group are all valid names for an R data object. Note also that R is case-sensitive, so agegroup and AgeGroup would be treated as different data objects.

To save the value 28 to a variable (data object) labelled age, run the code:

```r
age <- 28
```

- Inspecting a data object

To inspect the contents of the data object age run the following line of code:

```r
age
```

```
## [1] 28
```

Find out what kind (class) of data object age is using:

```r
class(age)
```

```
## [1] "numeric"
```

Inspect the structure of the age data object:

```r
str(age)
```
The vector data object

What if we have more than one response? We can use the \texttt{c( )} function to combine multiple values into one data vector object:

\begin{verbatim}
age <- c(28, 36, 25, 24, 32)
age
\end{verbatim}

\begin{verbatim}
# [1] 28 36 25 24 32
class(age) # Still numeric..
# [1] "numeric"
str(age) # . . . but now a vector (set) of 5 separate values
# num [1:5] 28 36 25 24 32
\end{verbatim}

Note that on each line in the code above any text following the \# character is ignored by R when executing the code. Instead, text following a \# can be used to add comments to the code to make clear what the code is doing. Two marks of good code are a clear layout and clear commentary on the code.

3.6.1 Basic Data Types

There are a number of data types. Four are the most common. In R, \texttt{numeric} is the default type for numbers. It stores all numbers as floating-point numbers (numbers with decimals). This is because most statistical calculations deal with numbers with up to two decimals.

- Numeric

\begin{verbatim}
num <- 4.5 # Decimal values
class(num)
\end{verbatim}

\begin{verbatim}
# [1] "numeric"
\end{verbatim}

- Integer

\begin{verbatim}
int <- as.integer(4) # Natural numbers. Note integers are also numerics.
class(int)
\end{verbatim}

\begin{verbatim}
# [1] "integer"
\end{verbatim}

- Character

\begin{verbatim}
cha <- "are you enjoying this?" # text or string. You can also type `as.character("are you enjoying this")`
class(cha)
\end{verbatim}

\begin{verbatim}
# [1] "character"
\end{verbatim}

- Logical
log <- 2 < 1 # assigns TRUE or FALSE. In this case, FALSE as 2 is greater than 1
log

## [1] FALSE
class(log)

## [1] "logical"

### 3.6.2 Random Variables

In statistics, we differentiate between data to capture:

- **Qualitative attributes** categorise objects eg. gender, marital status. To measure these attributes, we use *Categorical* data which can be divided into:
  - *Nominal* data in categories that have no inherent order eg. gender
  - *Ordinal* data in categories that have an inherent order eg. income bands

- **Quantitative attributes**:
  - *Discrete* data: count objects of a certain category eg. number of kids, cars
  - *Continuous* data: precise numeric measures eg. weight, income, length.

In R these three types of random variables are represented by the following types of R data object:

<table>
<thead>
<tr>
<th>variables</th>
<th>objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>nominal</td>
<td>factor</td>
</tr>
<tr>
<td>ordinal</td>
<td>ordered factor</td>
</tr>
<tr>
<td>discrete</td>
<td>numeric</td>
</tr>
<tr>
<td>continuous</td>
<td>numeric</td>
</tr>
</tbody>
</table>

We have already encountered the R data type *numeric*. The next section introduces the *factor* data type.

#### 3.6.2.1 Factor

**What is a factor?**

A factor variable assigns a numeric code to each possible category (*level*) in a variable. Behind the scenes, R stores the variable using these numeric codes to save space and speed up computing. For example, compare the size of a list of 10,000 *males* and *females* to a list of 10,000 1s and 0s. At the same time R also saves the category names associated with each numeric code (level). These are used for display purposes.
For example, the variable gender, converted to a factor, would be stored as a series of 1s and 2s, where 1 = female and 2 = male; but would be displayed in all outputs using their category labels of female and male.

### Creating a factor
To convert a numeric or character vector into a factor use the `factor()` function. For instance:

```r
gender <- c("female","male","male","female","female") # create a gender variable
gender <- factor(gender) # replace character vector with a factor version
gender
```

```r
# [1] female male male female female
# Levels: female male
```

```r
class(gender)
# [1] "factor"
```

```r
str(gender)
# Factor w/ 2 levels "female","male": 1 2 2 1 1
```

Now gender is a factor and is stored as a series of 1s and 2s, with 1s representing females and 2s representing males. The function `levels()` lists the levels (categories) associated with a given factor variable:

```r
levels(gender)
# [1] "female" "male"
```

The categories are reported in the order that they have been numbered (starting from 1). Hence from the output we can infer that females are coded as 1, and males as 2.

### 3.7 Data Frames

R stores different types of data using different types of data structure. Data are normally stored as a `data.frame`. A data frames contain one row per observation (e.g. wards) and one column per attribute (e.g. population and health).

We create three variables wards, population (`pop`) and people with good health (`ghealth`). We use 2011 census data counts for total population and good health for wards in Liverpool.

```r
wards <- c("Allerton and Hunts Cross","Anfield","Belle Vale","Central","Childwall","Church","Clubmoor",
           "County","Cressington","Croxteth","Everton","Fazakerley","Greenbank","Kensington and Fairfield",
           "Kirkdale","Knotty Ash","Mossley Hill","Norris Green","Old Swan","Picton","Princes Park",
           "Riverside","St Michael's","Speke-Garston","Tuebrook and Stoneycroft","Warbreck","Wavertree",
           "West Derby","Woolton","Yew Tree")
```

```r
pop <- c(14853,14510,15004,20340,13908,13974,15272,14045,14503,14561,14782,16786,16132,15377,16115,13312,
         13816,15047,16461,17009,17104,18422,12991,20300,16489,16481,14772,
         14651,14782,16786,16132,15377,16115,13312,13816,15047,16461,17009,17104,18422,12991,20300,16489,16481,14772
```

```r
```

```r
```
Note that `pop` and `ghealth` and `wards` contains characters.

### 3.7.1 Creating A Data Frame

We can create a data frame and examine its structure:

```r
df <- data.frame(wards, pop, ghealth)
df # or use `view(data)`
```

<table>
<thead>
<tr>
<th>wards</th>
<th>pop</th>
<th>ghealth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allerton and Hunts Cross</td>
<td>14853</td>
<td>7274</td>
</tr>
<tr>
<td>Anfield</td>
<td>14510</td>
<td>6124</td>
</tr>
<tr>
<td>Belle Vale</td>
<td>15004</td>
<td>6129</td>
</tr>
<tr>
<td>Central</td>
<td>20340</td>
<td>11925</td>
</tr>
<tr>
<td>Childwall</td>
<td>13908</td>
<td>7219</td>
</tr>
<tr>
<td>Church</td>
<td>13974</td>
<td>7461</td>
</tr>
<tr>
<td>Clubmoor</td>
<td>15272</td>
<td>6403</td>
</tr>
<tr>
<td>County</td>
<td>14045</td>
<td>5930</td>
</tr>
<tr>
<td>Cressington</td>
<td>14503</td>
<td>7094</td>
</tr>
<tr>
<td>Croxteth</td>
<td>14561</td>
<td>6992</td>
</tr>
<tr>
<td>Everton</td>
<td>14782</td>
<td>5517</td>
</tr>
<tr>
<td>Fazakerley</td>
<td>16786</td>
<td>7879</td>
</tr>
<tr>
<td>Greenbank</td>
<td>16132</td>
<td>8990</td>
</tr>
<tr>
<td>Kensington and Fairfield</td>
<td>15377</td>
<td>6495</td>
</tr>
<tr>
<td>Kirkdale</td>
<td>16115</td>
<td>6662</td>
</tr>
<tr>
<td>Knotty Ash</td>
<td>13312</td>
<td>5981</td>
</tr>
<tr>
<td>Mossley Hill</td>
<td>13816</td>
<td>7322</td>
</tr>
<tr>
<td>Norris Green</td>
<td>15047</td>
<td>6529</td>
</tr>
<tr>
<td>Old Swan</td>
<td>16461</td>
<td>7192</td>
</tr>
<tr>
<td>Picton</td>
<td>17009</td>
<td>7953</td>
</tr>
<tr>
<td>Princes Park</td>
<td>17104</td>
<td>7636</td>
</tr>
<tr>
<td>Riverside</td>
<td>18422</td>
<td>9001</td>
</tr>
<tr>
<td>St Michael's</td>
<td>12991</td>
<td>6450</td>
</tr>
<tr>
<td>Speke-Garston</td>
<td>20300</td>
<td>8973</td>
</tr>
<tr>
<td>Tuebrook and Stoneycroft</td>
<td>16489</td>
<td>7302</td>
</tr>
<tr>
<td>Warbreck</td>
<td>16481</td>
<td>7521</td>
</tr>
<tr>
<td>Wavertree</td>
<td>14772</td>
<td>7268</td>
</tr>
<tr>
<td>West Derby</td>
<td>14382</td>
<td>7013</td>
</tr>
<tr>
<td>Woolton</td>
<td>12921</td>
<td>6025</td>
</tr>
<tr>
<td>Yew Tree</td>
<td>16746</td>
<td>7717</td>
</tr>
</tbody>
</table>
3.7. DATA FRAMES

str(df) # or use glimpse(data)

## 'data.frame': 30 obs. of 3 variables:
##  $ wards: chr "Allerton and Hunts Cross" "Anfield" "Belle Vale" "Central" ... 
##  $ pop : num 14853 14510 15004 20340 13908 ... 
##  $ ghealth: num 7274 6124 6129 11925 7219 ... 

3.7.2 Referencing Data Frames

Throughout this module, you will need to refer to particular parts of a dataframe - perhaps a particular column (an area attribute); or a particular subset of respondents. Hence it is worth spending some time now mastering this particular skill.

The relevant R function, [ ], has the format [row,col] or, more generally, [set of rows, set of cols].

Run the following commands to get a feel of how to extract different slices of the data:

df # whole data.frame
df[1, 1] # contents of first row and column
df[2, 2:3] # contents of the second row, second and third columns
df[1, ] # first row, ALL columns [the default if no columns specified]
df[ ,1:2] # ALL rows; first and second columns
df[c(1,3,5), ] # rows 1,3,5; ALL columns
df[ , 2] # ALL rows; second column (by default results containing only one column are converted back into a vector)
df[ , 2, drop=FALSE] # ALL rows; second column (returned as a data.frame)

In the above, note that we have used two other R functions:

- 1:3 The colon operator tells R to produce a list of numbers including the named start and end points.
- c(1,3,5) Tells R to combine the contents within the brackets into one list of objects

Run both of these functions on their own to get a better understanding of what they do.

Three other methods for referencing the contents of a data.frame make direct use of the variable names within the data.frame, which tends to make for easier to read/understand code:

df[,"pop"] # variable name in quotes inside the square brackets
df$pop # variable name prefixed with $ and appended to the data.frame name
# or you can use attach
attach(df)
pop # but be careful if you already have an age variable in your local workspace
CHAPTER 3. DATA WRANGLING

Want to check the variables available, use the `names()`:

```r
names(df)
```

```r
## [1] "wards"  "pop"    "ghealth"
```

## 3.8 Read Data

### Ensure your memory is clear

```r
rm(list=ls()) # rm for targeted deletion / ls for listing all existing objects
```

There are many commands to read / load data onto R. The command to use will depend upon the format they have been saved. Normally they are saved in `csv` format from Excel or other software packages. So we use either:

- `df <- read.table("path/file_name.csv", header = FALSE, sep = ",")`
- `df <- read("path/file_name.csv", header = FALSE)`
- `df <- read.csv2("path/file_name.csv", header = FALSE)`

To read files in other formats, refer to this useful DataCamp tutorial

```r
census <- read.csv("data/census/census_data.csv")
```

```r
head(census)
```

```r
## # NOTE: always ensure your are setting the correct directory leading to the data.  # It may differ from your existing working directory
```

## 3.8.1 Quickly inspect the data

1. What class?

```r
# 1
class(census)
```

2. What R data types?

```r
# 2 & 3
str(census)
```
3.9. MANIPULATION DATA

Just interested in the variable names:

```r	names(census)
```

```r
# [1] "code" "ward" "pop16_74"
# [4] "higher_managerial" "pop" "ghealth"
```

or want to view the data:

```r
View(census)
```

3.9 Manipulation Data

3.9.1 Adding New Variables

Usually you want to add / create new variables to your data frame using existing variables eg. computing percentages by dividing two variables. There are many ways in which you can do this i.e. referencing a data frame as we have done above, or using $ (e.g. census$pop). For this module, we’ll use tidyverse:

```r
census <- census %>% mutate(per_ghealth = ghealth / pop)
```

Note we used a pipe operator %>% , which helps make the code more efficient and readable - more details, see Grolemund and Wickham (2019). When using the pipe operator, recall to first indicate the data frame before %>%. Note also the use a variable name before the = sign in brackets to indicate the name of the new variable after `mutate`.

3.9.2 Selecting Variables

Usually you want to select a subset of variables for your analysis as storing to large data sets in your R memory can reduce the processing speed of your machine. A selection of data can be achieved by using the `select` function:

```r
ndf <- census %>% select(ward, pop16_74, per_ghealth)
```

Again first indicate the data frame and then the variable you want to select to build a new data frame. Note the code chunk above has created a new data frame called `ndf`. Explore it.

3.9.3 Filtering Data

You may also want to filter values based on defined conditions. You may want to filter observations greater than a certain threshold or only areas within a certain region. For example, you may want to select areas with a percentage of good health population over 50%:

```r
ndf2 <- census %>% filter(per_ghealth < 0.5)
```
You can use more than one variables to set conditions. Use “,” to add a condition.

### 3.9.4 Joining Data Frames

When working with spatial data, we often need to join data. To this end, you need a common unique id variable. Let’s say, we want to add a data frame containing census data on households for Liverpool, and join the new attributes to one of the existing data frames in the workspace. First we will read the data frame we want to join (ie. `census_data2.csv`).

```r
# read data
census2 <- read.csv("data/census/census_data2.csv")
# visualise data structure
str(census2)
```

```r
## 'data.frame': 30 obs. of 3 variables:
## $ geo_code : chr "E05000886" "E05000887" "E05000888" "E05000889" ... 
## $ households : int 6359 6622 6622 7139 5391 5884 6576 6745 6317 6024 ... 
## $ socialrented_households: int 827 1508 2818 1311 374 178 2859 1564 1023 1558 ... 
```

The variable `geo_code` in this data frame corresponds to the `code` in the existing data frame and they are unique so they can be automatically matched by using the `merge()` function. The `merge()` function uses two arguments: `x` and `y`. The former refers to data frame 1 and the latter to data frame 2. Both of these two data frames must have a id variable containing the same information. Note they can have different names. Another key argument to include is `all.x=TRUE` which tells the function to keep all the records in `x`, but only those in `y` that match in case there are discrepancies in the id variable.

```r
# join data frames
join_dfs <- merge(census, census2, by.x="code", by.y="geo_code", all.x = TRUE)
# check data
head(join_dfs)
```

```r
## code ward pop16_74 higher_managerial pop ghealth
## 1 E05000886 Allerton and Hunts Cross 10930 1103 14853 7274
## 2 E05000887 Anfield 10712 312 14510 6124
## 3 E05000888 Belle Vale 10987 432 15004 6129
## 4 E05000889 Central 19174 1346 20340 11925
## 5 E05000890 Childwall 10410 1123 13908 7219
## 6 E05000891 Church 10569 1843 13974 7461
```

```r
## per_ghealth households socialrented_households
## 1  0.4897327 6359 827
## 2  0.4220538 6622 1508
## 3  0.4084911 6622 2818
## 4  0.5862832 7139 1311
## 5  0.5190538 5391 374
```
3.9.5 Saving Data

It may also be convenient to save your R projects. They contain all the objects that you have created in your workspace by using the `save.image()` function:

```
save.image("week1_envs453.RData")
```

This creates a file labelled “week1_envs453.RData” in your working directory. You can load this at a later stage using the `load()` function.

```
load("week1_envs453.RData")
```

Alternatively you can save / export your data into a csv file. The first argument in the function is the object name, and the second: the name of the csv we want to create.

```
write.csv(join_dfs, "join_censusdfs.csv")
```

3.10 Using Spatial Data Frames

A core area of this module is learning to work with spatial data in R. R has various purposely designed packages for manipulation of spatial data and spatial analysis techniques. Various R packages exist in CRAN eg. spatial, sgeostat, splancs, maptools, tmap, rgdal, spand and more recent development of sf - see Lovelace et al. (2020) for a great description and historical context for some of these packages.

During this session, we will use sf.

We first need to import our spatial data. We will use a shapefile containing data at Output Area (OA) level for Liverpool. These data illustrate the hierarchical structure of spatial data.

3.10.1 Read Spatial Data

```
oa_shp <- st_read("data/census/Liverpool_OA.shp")
```

```
## Reading layer `Liverpool_OA' from data source `/home/jovyan/work/data/census/Liverpool_OA.shp'
## Simple feature collection with 1584 features and 18 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox: xmin: 332390.2 ymin: 379748.5 xmax: 345636 ymax: 397980.1
## projected CRS: Transverse_Mercator
```

Examine the input data. A spatial data frame stores a range of attributes derived from a shapefile including the `geometry` of features (e.g. polygon shape
and location), attributes for each feature (stored in the .dbf), projection and coordinates of the shapefile’s bounding box - for details, execute:

```
?st_read
```

You can employ the usual functions to visualise the content of the created data frame:

```
# visualise variable names
names(oa_shp)
```

```
## [1] "OA_CD" "LSOA_CD" "MSOA_CD" "LAD_CD" "pop" "H_Vbad"
## [7] "H_bad" "H_fair" "H_good" "H_Vgood" "age_men" "age_med"
## [13] "age_60" "S_Rent" "Ethnic" "illness" "unemp" "males"
## [19] "geometry"
```

```
# data structure
str(oa_shp)
```

```
## Classes 'sf' and 'data.frame': 1584 obs. of 19 variables:
## $ OA_CD : chr "E00176737" "E00033515" "E00033141" "E00176757" ...  
## $ LSOA_CD : chr "E01033761" "E01006614" "E01006546" "E01006646" ...  
## $ MSOA_CD : chr "E02006932" "E02001358" "E02001365" "E02001369" ...  
## $ LAD_CD : chr "E08000012" "E08000012" "E08000012" "E08000012" ...  
## $ pop : int 185 281 208 200 321 187 395 320 316 214 ...  
## $ H_Vbad : int 1 2 3 7 4 4 5 9 5 4 ...  
## $ H_bad : int 2 20 10 8 10 25 19 22 25 17 ...  
## $ H_fair : int 9 47 22 17 32 70 42 53 55 39 ...  
## $ H_good : int 53 111 71 52 32 70 42 53 55 39 ...  
## $ H_Vgood : int 120 101 102 116 163 31 198 132 127 101 ...  
## $ age_men : num 27.9 37.7 37.1 33.7 34.2 ...  
## $ age_med : num 25 36 32 29 34 53 30 34 29 ...  
## $ age_60 : num 0.0108 0.1637 0.1971 0.1 0.1402 ...  
## $ S_Rent : num 0.0526 0.176 0.0235 0.2222 0.0222 ...  
## $ Ethnic : num 0.3514 0.0463 0.0192 0.215 0.0779 ...  
## $ illness : int 185 281 208 200 321 187 395 320 316 214 ...  
## $ unemp : num 0.0438 0.121 0.1121 0.036 0.0743 ...  
## $ males : int 122 128 95 120 158 123 207 164 157 94 ...  
## $ geometry:sfc_MULTIPOLYGON of length 1584; first list element: List of 1 
## ..$:List of 1 
## ...$: : num [1:14, 1:2] 335106 335130 335164 335173 335185 ...  
## ...- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
## - attr(*, "sf_column")= chr "geometry"  
## - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",...: NA NA NA NA NA NA NA NA NA NA NA
## ..- attr(*, "names")= chr [1:18] "OA_CD" "LSOA_CD" "MSOA_CD" "LAD_CD" ...  
```

```
# see first few observations
head(oa_shp)
```
3.10. USING SPATIAL DATA FRAMES

## Simple feature collection with 6 features and 18 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox: xmin: 335071.6 ymin: 389876.7 xmax: 339426.9 ymax: 394479
## projected CRS: Transverse_Mercator
## DA_CD LSOA_CD MSOA_CD LAD_CD pop H_Vbad H_bad H_fair H_good
## 1 E00176737 E01033761 E02006932 E08000012 185 1 2 9 53
## 2 E00033515 E01006614 E02001358 E08000012 281 2 20 47 111
## 3 E00033141 E01006546 E02001365 E08000012 208 3 10 22 71
## 4 E00176757 E01006646 E02001369 E08000012 200 7 8 17 52
## 5 E00034050 E01006712 E02001375 E08000012 321 4 10 32 112
## 6 E00034280 E01006761 E02001366 E08000012 187 4 25 70 57
## H_Vgood age_men age_med age_60 S_Rent Ethnic illness unemp
## 1 120 27.94054 25 0.01081081 0.05263158 0.35135135 185 0.04379562
## 2 101 37.71174 36 0.16370107 0.17600000 0.04626335 281 0.12101911
## 3 102 37.08173 32 0.19711538 0.02352941 0.01923077 208 0.11214953
## 4 116 33.73000 29 0.10000000 0.22222222 0.21500000 200 0.03597122
## 5 163 34.19003 34 0.14018692 0.02222222 0.07788162 321 0.07428571
## 6 31 56.09091 53 0.44919786 0.88524590 0.11764706 187 0.44615385
## males geometry
## 1 122 MULTIPOLYGON (((335106.3 38...}
## 2 128 MULTIPOLYGON (((335810.5 39...}
## 3 95 MULTIPOLYGON (((336738 3931...}
## 4 120 MULTIPOLYGON (((335914.5 39...}
## 5 158 MULTIPOLYGON (((339325 3914...}
## 6 123 MULTIPOLYGON (((338198.1 39...}

**TASK:**

- What are the geographical hierarchy in these data?
- What is the smallest geography?
- What is the largest geography?

### 3.10.2 Basic Mapping

Again, many functions exist in CRAN for creating maps:

- `plot` to create static maps
- `tmap` to create static and interactive maps
- `leaflet` to create interactive maps
- `mapview` to create interactive maps
- `ggplot2` to create data visualisations, including static maps
- `shiny` to create web applications, including maps

Here this notebook demonstrates the use of `plot` and `tmap`. First `plot` is used to map the spatial distribution of non-British-born population in Liverpool. First we only map the geometries on the right,
3.10.2.1 Using `plot`

```r
# mapping geometry
plot(st_geometry(oa_shp))
```

Figure 3.5: OAs of Liverpool

and then:

```r
# map attributes, adding intervals
plot(oa_shp[, "Ethnic"],
     key.pos = 4,
     axes = TRUE,
     key.width = lcm(1.3),
     key.length = 1.
     breaks = "jenks",
     lwd = 0.1,
     border = 'grey')
```

**TASK:**

- What is the key pattern emerging from this map?

3.10.2.2 Using `tmap`

Similar to `ggplot2`, `tmap` is based on the idea of a ‘grammar of graphics’ which involves a separation between the input data and aesthetics (i.e. the way data are visualised). Each data set can be mapped in various different ways, including location as defined by its geometry, colour and other features. The basic building block is `tm_shape()` (which defines input data), followed by one or more layer elements such as `tm_fill()` and `tm_dots()`.
Figure 3.6: Spatial distribution of ethnic groups, Liverpool

```r
# ensure geometry is valid
oa_shp <- sf::st_make_valid(oa_shp)

# map
legend_title <- expression("% ethnic pop.")
map_oa <- tm_shape(oa_shp) +
  tm_fill(col = "Ethnic", title = legend_title, palette = magma(256), style = "cont") + # add fill
  tm_borders(col = "white", lwd = .01) + # add borders
  tm_compass(type = "arrow", position = c("right", "top"), size = 4) + # add compass
  tm_scale_bar(breaks = c(0,1,2), text.size = 0.5, position = c("center", "bottom")) # add scale
map_oa
```
Note that the operation + is used to add new layers. You can set style themes by tm_style. To visualise the existing styles use tmap_style_catalogue(), and you can also evaluate the code chunk below if you would like to create an interactive map.

```r
# Read data at the OA level
msoa_shp <- st_read("data/census/Liverpool_MSOA.shp")
```

**TASK:**
- Try mapping other variables in the spatial data frame. Where do population aged 60 and over concentrate?

### 3.10.3 Comparing geographies

If you recall, one of the key issues of working with spatial data is the modifiable area unit problem (MAUP) - see lecture notes. To get a sense of the effects of MAUP, we analyse differences in the spatial patterns of the ethnic population in Liverpool between Middle Layer Super Output Areas (MSOAs) and OAs. So we map these geographies together.

```r
# Read data at the msoa level
msoa_shp <- st_read("data/census/Liverpool_MSOA.shp")
```

## Reading layer 'Liverpool_MSOA' from data source '/home/jovyan/work/data/census/Liverpool_MSOA.shp' using driver 'ESRI Shapefile'

## Simple feature collection with 61 features and 16 fields
## geometry type: MULTIPOLYGON
3.11. USEFUL FUNCTIONS

## dimension: XY
## bbox: xmin: 333086.1 ymin: 381426.3 xmax: 345636 ymax: 397980.1
## projected CRS: Transverse_Mercator

# ensure geometry is valid
msoa_shp = sf::st_make_valid(msoa_shp)

# create a map
map_msoa = tm_shape(msoa_shp) +
  tm_fill(col = "Ethnic", title = legend_title, palette = magma(256), style = "cont") +
  tm_borders(col = "white", lwd = .01) +
  tm_compass(type = "arrow", position = c("right", "top"), size = 4) +
  tm_scale_bar(breaks = c(0,1,2), text.size = 0.5, position = c("center", "bottom"))

# arrange maps
map_arrange(map_msoa, map_oa)

---

**TASK:**

- What differences do you see between OAs and MSOAs?
- Can you identify areas of spatial clustering? Where are they?

### 3.11 Useful Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read.csv()</td>
<td>read csv files into data frames</td>
</tr>
<tr>
<td>str()</td>
<td>inspect data structure</td>
</tr>
<tr>
<td>mutate()</td>
<td>create a new variable</td>
</tr>
<tr>
<td>filter()</td>
<td>filter observations based on variable values</td>
</tr>
<tr>
<td>%&gt;%</td>
<td>pipe operator - chain operations</td>
</tr>
<tr>
<td>select()</td>
<td>select variables</td>
</tr>
<tr>
<td>merge()</td>
<td>join dat frames</td>
</tr>
<tr>
<td>st_read</td>
<td>read spatial data (ie. shapefiles)</td>
</tr>
<tr>
<td>plot()</td>
<td>create a map based a spatial data set</td>
</tr>
<tr>
<td>tm_shape(), tm_fill(), tmBorders()</td>
<td>create a map using tmap functions</td>
</tr>
<tr>
<td>tm_arrange</td>
<td>display multiple maps in a single “metaplot”</td>
</tr>
</tbody>
</table>
Chapter 4

Points

This chapter is based on the following references, which are great follow-up’s on the topic:

- Lovelace and Cheshire (2014) is a great introduction.
- Chapter 6 of Brunsdon and Comber (2015), in particular subsections 6.3 and 6.7.
- Bivand et al. (2013) provides an in-depth treatment of spatial data in R.

4.1 Dependencies

We will rely on the following libraries in this section, all of them included in the book list:

```r
# For pretty table
library(knitr)
# All things geodata
library(sf)

## Linking to GEOS 3.8.0, GDAL 3.0.4, PROJ 6.3.1
library(sp)
# Pretty graphics
library(ggplot2)
library(gridExtra)
# Thematic maps
library(tmap)
# Pretty maps
library(ggmap)

## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```
Before we start any analysis, let us set the path to the directory where we are working. We can easily do that with \texttt{setwd()}. Please replace in the following line the path to the folder where you have placed this file-and where the \texttt{house\_transactions} folder with the data lives.

\begin{verbatim}
setwd('.'
\end{verbatim}

## 4.2 Data

For this session, we will use the set of Airbnb properties for San Diego (US), borrowed from the “Geographic Data Science with Python” book (see here for more info on the dataset source). This covers the point location of properties advertised on the Airbnb website in the San Diego region.

Let us start by reading the data, which comes in a GeoJSON:

\begin{verbatim}
db <- st_read("data/abb_sd/regression_db.geojson")
\end{verbatim}

We can then examine the columns of the table with the \texttt{colnames} method:

\begin{verbatim}
colnames(db)
\end{verbatim}

The rest of this session will focus on two main elements of the table: the spatial dimension (as stored in the point coordinates), and the nightly price values, expressed in USD and contained in the \texttt{price} column. To get a sense of what they look like first, let us plot both. We can get a quick look at the non-spatial distribution of house values with the following commands:
# Create the histogram

```r
hist <- qplot(data=db, x=price)
hist
```

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

![Histogram of Raw AirBnb prices in San Diego](image)

**Figure 4.1: Raw AirBnb prices in San Diego**

This basically shows there is a lot of values concentrated around the lower end of the distribution but a few very large ones. A usual transformation to shrink these differences is to take logarithms. The original table already contains an additional column with the logarithm of each price (`log_price`).

```r
# Create the histogram
hist <- qplot(data=db, x=log_price)
hist
```

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

To obtain the spatial distribution of these houses, we need to focus on the `geometry` column. The easiest, quickest (and also “dirtiest”) way to get a sense of what the data look like over space is using `plot`:

```r
plot(st_geometry(db))
```

Now this has the classic problem of cluttering: some portions of the map have
so many points that we can’t tell what the distribution is like. To get around this issue, there are two solutions: binning and smoothing.

### 4.3 Binning

The two-dimensional sister of histograms are binning maps: we divide each of the two dimensions into “buckets”, and count how many points fall within each bucket. Unlike histograms, we encode that count with a color gradient rather than a bar chart over an additional dimension (for that, we would need a 3D plot). These “buckets” can be squares (left) or hexagons (right):

```r
# Squared binning
# Set up plot
sqbin <- ggplot() +
# Add 2D binning with the XY coordinates as
# a dataframe
geom_bin2d(  
data=as.data.frame(st_coordinates(db)),
aes(x=X, y=Y)
)

# Hex binning
```
4.3. BINNING

Figure 4.3: Spatial distribution of AirBnb in San Diego

```r
# Set up plot
hexbin <- ggplot() +
# Add hex binning with the XY coordinates as a dataframe
geom_hex(
  data=as.data.frame(st_coordinates(db)),
  aes(x=X, y=Y)
) +
# Use viridis for color encoding (recommended)
scale_fill_continuous(type = "viridis")
# Bind in subplots
grid.arrange(sqbin, hexbin, ncol=2)
```
4.4 KDE

Kernel Density Estimation (KDE) is a technique that creates a continuous representation of the distribution of a given variable, such as house prices. Although theoretically it can be applied to any dimension, usually, KDE is applied to either one or two dimensions.

4.4.1 One-dimensional KDE

KDE over a single dimension is essentially a contiguous version of a histogram. We can see that by overlaying a KDE on top of the histogram of logs that we have created before:

```r
# Create the base
base <- ggplot(db, aes(x=log_price))
# Histogram
hist <- base +
  geom_histogram(bins=50, aes(y..density..))
# Overlay density plot
kde <- hist +
  geom_density(fill="#FF6666", alpha=0.5, colour="#FF6666")
kde
```

The key idea is that we are smoothing out the discrete binning that the histogram involves. Note how the histogram is exactly the same as above shape-
4.4. KDE

4.4.2 Two-dimensional (spatial) KDE

Geography, at the end of the day, is usually represented as a two-dimensional space where we locate objects using a system of dual coordinates, $X$ and $Y$ (or latitude and longitude). Thanks to that, we can use the same technique as above to obtain a smooth representation of the distribution of a two-dimensional variable. The crucial difference is that, instead of obtaining a curve as the output, we will create a surface, where intensity will be represented with a color gradient, rather than with the second dimension, as it is the case in the figure above.

To create a spatial KDE in R, we can use general tooling for non-spatial points, such as the `stat_density2d_filled` method:

```r
# Create the KDE surface
kde <- ggplot(data = db) +
  stat_density2d_filled(
    data = as.data.frame(st_coordinates(db)),
    aes(x = X, y = Y, alpha = ..level..),
```
n = 16

# Tweak the color gradient
scale_color_viridis_c()

# White theme
theme_bw()

# Tip! Add invisible points to improve proportions
kde + geom_sf(alpha=0)

Figure 4.5: KDE of AirBnb properties in San Diego

This approach generates a surface that represents the density of dots, that is an estimation of the probability of finding a house transaction at a given coordinate. However, without any further information, they are hard to interpret and link with previous knowledge of the area. To bring such context to the figure, we can plot an underlying basemap, using a cloud provider such as Google Maps or, as in this case, OpenStreetMap. To do it, we will leverage the library `ggmap`, which is designed to play nicely with the `ggplot2` family (hence the seemingly counterintuitive example above). Before we can plot them with the online map, we need to reproject them though.

# Reproject coordinates
lon_lat <- st_transform(db, crs = 4326) %>%
  st_coordinates() %>%
4.5 Spatial Interpolation

The previous section demonstrates how to visualize the distribution of a set of spatial objects represented as points. In particular, given a bunch of house locations, it shows how one can effectively visualize their distribution over space and get a sense of the density of occurrences. Such visualization, because it is based on KDE, is based on a smooth continuum, rather than on a discrete approach (as a choropleth would do, for example).
CHAPTER 4. POINTS

Many times however, we are not particularly interested in learning about the density of occurrences, but about the distribution of a given value attached to each location. Think for example of weather stations and temperature: the location of the stations is no secret and rarely changes, so it is not of particular interest to visualize the density of stations; what we are usually interested instead is to know how temperature is distributed over space, given we only measure it in a few places. One could argue the example we have been working with so far, house prices in AirBnb, fits into this category as well: although where a house is advertised may be of relevance, more often we are interested in finding out what the “surface of price” looks like. Rather than where are most houses being advertised? we usually want to know where the most expensive or most affordable houses are located.

In cases where we are interested in creating a surface of a given value, rather than a simple density surface of occurrences, KDE cannot help us. In these cases, what we are interested in is spatial interpolation, a family of techniques that aim at exactly that: creating continuous surfaces for a particular phenomenon (e.g. temperature, house prices) given only a finite sample of observations. Spatial interpolation is a large field of research that is still being actively developed and that can involve a substantial amount of mathematical complexity in or-
4.5. SPATIAL INTERPOLATION

...der to obtain the most accurate estimates possible\(^1\). In this chapter, we will introduce the simplest possible way of interpolating values, hoping this will give you a general understanding of the methodology and, if you are interested, you can check out further literature. For example, Banerjee et al. (2014) or Cressie (2015) are hard but good overviews.

4.5.1 Inverse Distance Weight (IDW) interpolation

The technique we will cover here is called Inverse Distance Weighting, or IDW for convenience. Brunsdon and Comber (2015) offer a good description:

> In the inverse distance weighting (IDW) approach to interpolation, to estimate the value of \( z \) at location \( x \) a weighted mean of nearby observations is taken [...]. To accommodate the idea that observations of \( z \) at points closer to \( x \) should be given more importance in the interpolation, greater weight is given to these points [...]

— Page 204

The math\(^2\) is not particularly complicated and may be found in detail elsewhere (the reference above is a good starting point), so we will not spend too much time on it. More relevant in this context is the intuition behind. The idea is that we will create a surface of house price by smoothing many values arranged along a regular grid and obtained by interpolating from the known locations to the regular grid locations. This will give us full and equal coverage to soundly perform the smoothing.

Enough chat, let’s code\(^3\).

From what we have just mentioned, there are a few steps to perform an IDW spatial interpolation:

1. Create a regular grid over the area where we have house transactions.
2. Obtain IDW estimates for each point in the grid, based on the values of \( k \) nearest neighbors.
3. Plot a smoothed version of the grid, effectively representing the surface of house prices.

Let us go in detail into each of them\(^4\). First, let us set up a grid for the extent of the bounding box of our data (not the use of pipe, \( %>% \), operator to chain...
functions):

```r
sd.grid <- db %>%
  st_bbox() %>%
  st_as_sfc() %>%
  st_make_grid(
    n = 100,
    what = "centers"
  ) %>%
  st_as_sf() %>%
  cbind(., st_coordinates(.))
```

The object `sd.grid` is a regular grid with 10,000 (100 × 100) equally spaced cells:

```
## Simple feature collection with 10000 features and 2 fields
## geometry type: POINT
## dimension: XY
## geographic CRS: WGS 84
## First 10 features:
##     X         Y
## 1 -117.2795 32.57604 POINT (-117.2795 32.57604)
## 2 -117.2763 32.57604 POINT (-117.2763 32.57604)
## 3 -117.2730 32.57604 POINT (-117.273 32.57604)
## 4 -117.2698 32.57604 POINT (-117.2698 32.57604)
## 5 -117.2665 32.57604 POINT (-117.2665 32.57604)
## 6 -117.2632 32.57604 POINT (-117.2632 32.57604)
## 7 -117.2600 32.57604 POINT (-117.26 32.57604)
## 8 -117.2567 32.57604 POINT (-117.2567 32.57604)
## 9 -117.2535 32.57604 POINT (-117.2535 32.57604)
##10 -117.2502 32.57604 POINT (-117.2502 32.57604)
```

Now, `sd.grid` only contain the location of points to which we wish to interpolate. That is, we now our “target” geography for which we’d like to have Airbnb prices, but we don’t have price estimates. For that, on to the IDW, which will generate estimates for locations in `sd.grid` based on the observed prices in `db`. Again, this is hugely simplified by `gstat`:

```r
idw.hp <- idw(
  price ~ 1,  # Formula for IDW
  locations = db,  # Initial locations with values
  newdata = sd.grid,  # Locations we want predictions for
  nmax = 150  # Limit the number of neighbours for IDW
)
```
Boom! We’ve got it. Let us pause for a second to see how we just did it. First, we pass `price ~ 1`. This specifies the formula we are using to model house prices. The name on the left of ~ represents the variable we want to explain, while everything to its right captures the explanatory variables. Since we are considering the simplest possible case, we do not have further variables to add, so we simply write `1`. Then we specify the original locations for which we do have house prices (our original `db` object), and the points where we want to interpolate the house prices (the `sd.grid` object we just created above). One more note: by default, `idw` uses all the available observations, weighted by distance, to provide an estimate for a given point. If you want to modify that and restrict the maximum number of neighbors to consider, you need to tweak the argument `nmax`, as we do above by using the 150 nearest observations to each point\(^5\).

The object we get from `idw` is another spatial table, just as `db`, containing the interpolated values. As such, we can inspect it just as with any other of its kind. For example, to check out the top of the estimated table:

```r
head(idw.hp)
```

```r
## Simple feature collection with 6 features and 2 fields
## geometry type: POINT
## dimension: XY
## bbox: xmin: -117.2795 ymin: 32.57604 xmax: -117.2632 ymax: 32.57604
## geographic CRS: WGS 84
## var1.pred var1.var geometry
## 1 295.6100 NA POINT (-117.2795 32.57604)
## 2 295.1651 NA POINT (-117.2763 32.57604)
## 3 296.5927 NA POINT (-117.2730 32.57604)
## 4 288.2252 NA POINT (-117.2698 32.57604)
## 5 281.5522 NA POINT (-117.2665 32.57604)
## 6 268.3567 NA POINT (-117.2632 32.57604)
```

The column we will pay attention to is `var1.pred`. For a hypothetical house advertised at the location in the first row of point in `sd.grid`, the price IDW would guess it would cost, based on prices nearby, is the first element of column `var1.pred` in `idw.hp`.

\(^5\)Have a play with this because the results do change significantly. Can you reason why?
4.5.2 A surface of housing prices

Once we have the IDW object computed, we can plot it to explore the distribution, not of AirBnb locations in this case, but of house prices over the geography of San Diego. To do this using \texttt{ggplot2}, we first append the coordinates of each grid cell as columns of the table:

\begin{verbatim}
idw.hp = idw.hp %>%
  cbind(st_coordinates(.))
\end{verbatim}

Now, we can visualise the surface using standard \texttt{ggplot2} tools:

\begin{verbatim}
ggplot(idw.hp, aes(x = X, y = Y, fill = var1.pred)) + geom_raster()
\end{verbatim}

And we can “dress it up” a bit further:

\begin{verbatim}
ggplot(idw.hp, aes(x = X, y = Y, fill = var1.pred)) + geom_raster() +
scale_fill_viridis_b() +
theme_void() + geom_sf(alpha=0)
\end{verbatim}
Looking at this, we can start to tell some patterns. To bring in context, it would be great to be able to add a basemap layer, as we did for the KDE. This is conceptually very similar to what we did above, starting by reprojecting the points and continuing by overlaying them on top of the basemap. However, technically speaking it is not possible because \texttt{ggmap} -- the library we have been using to display tiles from cloud providers -- does not play well with our own rasters (i.e. the price surface). At the moment, it is surprisingly tricky to get this to work, so we will park it for now\(^6\).

### 4.5.3 “What should the next house’s price be?”

The last bit we will explore in this session relates to prediction for new values. Imagine you are a real state data scientist working for Airbnb and your boss asks you to give an estimate of how much a new house going into the market should cost. The only information you have to make such a guess is the location of the house. In this case, the IDW model we have just fitted can help you. The trick is realizing that, instead of creating an entire grid, all we need is to obtain an estimate of a single location.

Let us say, a new house is going to be advertised on the coordinates \(X = -117.02259063720702, Y = 32.76511965117273\) as expressed in longitude and latitude. In that case, we can do as follows:

\(^6\) \textbf{BONUS} if you can figure out a way to do it yourself!
pt <- c(X = -117.02259063720702, Y = 32.76511965117273) %>%
st_point() %>%
st_sfc() %>%
st_sf(crs = "EPSG:4326")
idw.one <- idw(price ~ 1, locations=db, newdata=pt)

## Warning in proj4string(d$data): CRS object has comment, which is lost in output
## Warning in proj4string(newdata): CRS object has comment, which is lost in output
## [inverse distance weighted interpolation]
## Warning in proj4string(newdata): CRS object has comment, which is lost in output
idw.one

## Simple feature collection with 1 feature and 2 fields
## geometry type: POINT
## dimension: XY
## bbox: xmin: -117.0226 ymin: 32.76512 xmax: -117.0226 ymax: 32.76512
## geographic CRS: WGS 84
## var1.pred var1.var geometry
## 1 171.4141 NA POINT (-117.0226 32.76512)

And, as show above, the estimated value is $171.4141334\textsuperscript{7}$.

### 4.6 Questions

We will be using the Madrid AirBnb dataset:

```r
mad_abb <- st_read("data/assignment_1_madrid/madrid_abb.gpkg")
```

## Reading layer `madrid_abb' from data source `/home/jovyan/work/data/assignment_1_madrid/madrid_abb.gpkg' using driver `GPKG'
## Simple feature collection with 18399 features and 16 fields
## geometry type: POINT
## dimension: XY
## geographic CRS: WGS 84

colnames(mad_abb)

## [1] "price"       "price_usd"    "log1p_price_usd" "accommodates"
## [5] "bathrooms_text" "bathrooms"    "bedrooms"    "beds"
## [9] "neighbourhood" "room_type"    "property_type" "WiFi"

\textsuperscript{7}\textbf{PRO QUESTION} Is that house expensive or cheap, as compared to the other houses sold in this dataset? Can you figure out where the house is in the distribution?
For this set of questions, the only two columns we will need is geom, which contains the point geometries, and price_usd, which record the price of the AirBnb property in USD.

With this at hand, answer the following questions:

1. Create a KDE that represents the density of locations of AirBnb properties in Madrid
2. Using inverse distance weighting, create a surface of AirBnb prices
Chapter 5

Flows

This chapter covers spatial interaction flows. Using open data from the city of San Francisco about trips on its bikeshare system, we will estimate spatial interaction models that try to capture and explain the variation in the amount of trips on each given route. After visualizing the dataset, we begin with a very simple model and then build complexity progressively by augmenting it with more information, refined measurements, and better modeling approaches. Throughout the chapter, we explore different ways to grasp the predictive performance of each model. We finish with a prediction example that illustrates how these models can be deployed in a real-world application.

Content is based on the following references, which are great follow-up’s on the topic:

- Singleton (2017), an online short course on R for Geographic Data Science and Urban Analytics. In particular, the section on mapping flows is specially relevant here.
- The predictive checks section draws heavily from Gelman and Hill (2006a), in particular Chapters 6 and 7.

5.1 Dependencies

We will rely on the following libraries in this section, all of them included in the book list:

```r
# Spatial Data management
library(rgdal)
```

```
# Loading required package: sp

## rgdal: version: 1.5-23, (SVN revision 1121)
## Geospatial Data Abstraction Library extensions to R successfully loaded
```
In this chapter we will show a slightly different way of managing spatial data in R. Although most of the functionality will be similar to that seen in previous chapters, we will not rely on the "sf stack" and we will instead show how to read and manipulate data using the more traditional sp stack. Although this approach is being slowly phased out, it is still important to be aware of its existence and its differences with more modern approaches.

Before we start any analysis, let us set the path to the directory where we are working. We can easily do that with `setwd()`. Please replace in the following line the path to the folder where you have placed this file -and where the sf_bikes folder with the data lives.

```
setwd(".")
```
5.2. Data

In this note, we will use data from the city of San Francisco representing bike trips on their public bike share system. The original source is the SF Open Data portal (link) and the dataset comprises both the location of each station in the Bay Area as well as information on trips (station of origin to station of destination) undertaken in the system from September 2014 to August 2015 and the following year. Since this note is about modeling and not data preparation, a cleanly reshaped version of the data, together with some additional information, has been created and placed in the sf_bikes folder. The data file is named flows.geojson and, in case you are interested, the (Python) code required to created from the original files in the SF Data Portal is also available on the flows_prep.ipynb notebook [url], also in the same folder.

Let us then directly load the file with all the information necessary:

```r
db <- readOGR('./data/sf_bikes/flows.geojson')
```

## OGR data source with driver: GeoJSON
## Source: "/home/jovyan/work/data/sf_bikes/flows.geojson", layer: "flows"
## with 1722 features
## It has 9 fields

rownames(db$data) <- db$flow_id
db$data$flow_id <- NULL

Note how the interface is slightly different since we are reading a GeoJSON file instead of a shapefile.

The data contains the geometries of the flows, as calculated from the Google Maps API, as well as a series of columns with characteristics of each flow:

```r
head(db$data)
```

```
# dest orig straight_dist street_dist total_down total_up trips15 trips16
# 39-41 41 39 1452.201 1804.1150 11.205753 4.698162 68 68
# 39-42 42 39 1734.861 2069.1557 10.290236 2.897886 23 29
# 39-45 45 39 1255.349 1747.9928 11.015596 4.593927 83 50
# 39-46 46 39 715.689 769.9189 0.000000 3.282495 127 73
```

where `orig` and `dest` are the station IDs of the origin and destination, `street/straight_dist` is the distance in metres between stations measured along the street network or as-the-crow-flies, `total_down/up` is the total downhill and climb in the trip, and `tripsXX` contains the amount of trips undertaken in the years of study.
5.3 “Seeing” flows

The easiest way to get a quick preview of what the data looks like spatially is to make a simple plot:

```r
plot(db)
```

![Figure 5.1: Potential routes](image)

Equally, if we want to visualize a single route, we can simply subset the table. For example, to get the shape of the trip from station 39 to station 48, we can:

```r
one39to48 <- db[ which(
    db@data$orig == 39 & db@data$dest == 48
  ), ]
plot(one39to48)
```

or, for the most popular route, we can:

```r
most_pop <- db[ which(
    db@data$trips15 == max(db@data$trips15)
  ), ]
plot(most_pop)
```

These however do not reveal a lot: there is no geographical context (why are there so many routes along the NE?) and no sense of how volumes of bikers are
allocated along different routes. Let us fix those two.

The easiest way to bring in geographical context is by overlaying the routes on top of a background map of tiles downloaded from the internet. Let us download this using `ggmap`:

```r
sf_bb <- c(
    left=db@bbox['x', 'min'],
    right=db@bbox['x', 'max'],
    bottom=db@bbox['y', 'min'],
    top=db@bbox['y', 'max']
  )
SanFran <- get_stamenmap(
    sf_bb,
    zoom = 14,
    maptype = "toner-lite"
)
```

## Source : http://tile.stamen.com/toner-lite/14/2620/6330.png
## Source : http://tile.stamen.com/toner-lite/14/2621/6330.png
## Source : http://tile.stamen.com/toner-lite/14/2622/6330.png

Figure 5.2: Trip from station 39 to 48
Figure 5.3: Most popular trip

```r
# Source: http://tile.stamen.com/toner-lite/14/2620/6331.png
# Source: http://tile.stamen.com/toner-lite/14/2621/6331.png
# Source: http://tile.stamen.com/toner-lite/14/2622/6331.png
# Source: http://tile.stamen.com/toner-lite/14/2620/6332.png
# Source: http://tile.stamen.com/toner-lite/14/2621/6332.png
# Source: http://tile.stamen.com/toner-lite/14/2622/6332.png
# Source: http://tile.stamen.com/toner-lite/14/2620/6333.png
# Source: http://tile.stamen.com/toner-lite/14/2621/6333.png
# Source: http://tile.stamen.com/toner-lite/14/2622/6333.png
```
and make sure it looks like we intend it to look:

```r
ggmap(SanFran)
```
Now to combine tiles and routes, we need to pull out the coordinates that make up each line. For the route example above, this would be:

```r
xys1 <- as.data.frame(coordinates(most_pop))
```

Now we can plot the route\(^1\) (note we also dim down the background to focus the attention on flows):

```r
ggmap(SanFran, darken=0.5) +
geom_path(
  aes(x=X1, y=X2),
  data=xys1,
  size=1,
  color=rgb(0.996078431372549, 0.7019607843137254, 0.03137254901960784),
  lineend='round'
)
```

\(^1\) **EXERCISE**: can you plot the route for the largest climb?
Now we can plot all of the lines by using a short `for` loop to build up the table:

```r
# Set up shell data.frame
lines <- data.frame(
    lat = numeric(0),
    lon = numeric(0),
    trips = numeric(0),
    id = numeric(0)
)

# Run loop for(x in 1:nrow(db)){
    # Pull out row
    r <- db[x,]
    # Extract lon/lat coords
    xys <- as.data.frame(coordinates(r))
    names(xys) <- c('lon', 'lat')
    # Insert trips and id
    xys['trips'] <- r@data$trips15
    xys['id'] <- x
    # Append them to `lines`
    lines <- rbind(lines, xys)
}
```

Now we can go on and plot all of them:
Finally, we can get a sense of the distribution of the flows by associating a color gradient to each flow based on its number of trips:

```r
ggmap(SanFran, darken=0.75) +
geom_path(
  aes(x=lon, y=lat, group=id, colour=trips),
  data=lines,
  size=log1p(lines$trips / max(lines$trips)),
  lineend='round'
) +
scale_colour_gradient(
  low='grey', high='#07eda0'
) +
theme(
  axis.text.x = element_blank(),
  axis.text.y = element_blank(),
)```
Note how we transform the size so it’s a proportion of the largest trip and then it is compressed with a logarithm.

5.4 Modelling flows

Now we have an idea of the spatial distribution of flows, we can begin to think about modeling them. The core idea in this section is to fit a model that can capture the particular characteristics of our variable of interest (the volume of trips) using a set of predictors that describe the nature of a given flow. We will start from the simplest model and then progressively build complexity until we get to a satisfying point. Along the way, we will be exploring each model using concepts from Gelman and Hill (2006a) such as predictive performance checks\(^2\) (PPC)

Before we start running regressions, let us first standardize the predictors so we can interpret the intercept as the average flow when all the predictors take the average value, and so we can interpret the model coefficients as changes in standard deviation units:

\[
\textit{# Scale all the table}
\]

\[
db\_std \leftarrow \text{as.data.frame(scale(db@data))}
\]

\(^2\)For a more elaborate introduction to PPC, have a look at Chapters 7 and 8.
Baseline model

One of the simplest possible models we can fit in this context is a linear model that explains the number of trips as a function of the straight distance between the two stations and total amount of climb and downhill. We will take this as the baseline on which we can further build later:

```r
m1 <- lm('trips15 ~ straight_dist + total_up + total_down', data=db_std)
summary(m1)
```

To explore how good this model is, we will be comparing the predictions the model makes about the number of trips each flow should have with the actual number of trips. A first approach is to simply plot the distribution of both variables:

```r
plot(density(m1$fitted.values), 
     xlim=c(-100, max(db_std$trips15)), 
     main='')
```
The plot makes pretty obvious that our initial model captures very few aspects of the distribution we want to explain. However, we should not get too attached to this plot just yet. What it is showing is the distribution of predicted point estimates from our model. Since our model is not deterministic but inferential, there is a certain degree of uncertainty attached to its predictions, and that is completely absent from this plot.

Generally speaking, a given model has two sources of uncertainty: predictive, and inferential. The former relates to the fact that the equation we fit does not capture all the elements or in the exact form they enter the true data generating process; the latter has to do with the fact that we never get to know the true...
value of the model parameters only guesses (estimates) subject to error and
uncertainty. If you think of our linear model above as

\[ T_{ij} = X_{ij} \beta + \epsilon_{ij} \]

where \( T_{ij} \) represents the number of trips undertaken between station \( i \) and \( j \), 
\( X_{ij} \) is the set of explanatory variables (length, climb, descent, etc.), and \( \epsilon_{ij} \) is an error term assumed to be distributed as a normal distribution \( N(0, \sigma) \); then predictive uncertainty comes from the fact that there are elements to some extent relevant for \( y \) that are not accounted for and thus subsumed into \( \epsilon_{ij} \). Inferential uncertainty comes from the fact that we never get to know \( \beta \) but only an estimate of it which is also subject to uncertainty itself.

Taking these two sources into consideration means that the black line in the plot above represents only the behaviour of our model we expect if the error term is absent (no predictive uncertainty) and the coefficients are the true estimates (no inferential uncertainty). However, this is not necessarily the case as our estimate for the uncertainty of the error term is certainly not zero, and our estimates for each parameter are also subject to a great deal of inferential variability. we do not know to what extent other outcomes would be just as likely. Predictive checking relates to simulating several feasible scenarios under our model and use those to assess uncertainty and to get a better grasp of the quality of our predictions.

Technically speaking, to do this, we need to build a mechanism to obtain a possible draw from our model and then repeat it several times. The first part of those two steps can be elegantly dealt with by writing a short function that takes a given model and a set of predictors, and produces a possible random draw from such model:

```r
generate_draw <- function(m){
  # Set up predictors matrix
  x <- model.matrix(m)
  # Obtain draws of parameters (inferential uncertainty)
  sim_bs <- sim(m, 1)
  # Predicted value
  mu <- x %*% sim_bs@coef[1, ]
  # Draw
  n <- length(mu)
  y_hat <- rnorm(n, mu, sim_bs@sigma[1])
  return(y_hat)
}
```

This function takes a model \( m \) and the set of covariates \( x \) used and returns a random realization of predictions from the model. To get a sense of how this works, we can get and plot a realization of the model, compared to the expected one and the actual values:
new_y <- generate_draw(m1)

plot(
  density(m1$fitted.values),
  xlim=c(-100, max(db_std$trips15)),
  ylim=c(0, max(c(
    max(density(m1$fitted.values)$y),
    max(density(db_std$trips15)$y)
  )
  ),
  col='black',
  main=''
)
lines(
  density(db_std$trips15),
  col='red',
  main='
)
lines(
  density(new_y),
  col='green',
  main='
)
legend(
  'topright',
  c('Predicted', 'Actual', 'Simulated'),
  col=c('black', 'red', 'green'),
  lwd=1
)
Once we have this “draw engine”, we can set it to work as many times as we want using a simple for loop. In fact, we can directly plot these lines as compared to the expected one and the trip count:

```r
# Loop for realizations
for(i in 1:250){
  tmp_y <- generate_draw(m1)
  lines(density(tmp_y),
        col='grey',
        lwd=0.1
  )
}
```

![Graph showing density predictions and actual trip count](image)
The plot shows there is a significant mismatch between the fitted values, which are much more concentrated around small positive values, and the realizations of our “inferential engine”, which depict a much less concentrated distribution of values. This is likely due to the combination of two different reasons: on the one hand, the accuracy of our estimates may be poor, causing them to jump around a wide range of potential values and hence resulting in very diverse predictions (inferential uncertainty); on the other hand, it may be that the
amount of variation we are not able to account for in the model is so large that the degree of uncertainty contained in the error term of the model is very large, hence resulting in such a flat predictive distribution.

It is important to keep in mind that the issues discussed in the paragraph above relate only to the uncertainty behind our model, not to the point predictions derived from them, which are a mechanistic result of the minimization of the squared residuals and hence are not subject to probability or inference. That allows them in this case to provide a fitted distribution much more accurate apparently (black line above). However, the lesson to take from this model is that, even if the point predictions (fitted values) are artificially accurate, our capabilities to infer about the more general underlying process are fairly limited.

Improving the model

The bad news from the previous section is that our initial model is not great at explaining bike trips. The good news is there are several ways in which we can improve this. In this section we will cover three main extensions that exemplify three different routes you can take when enriching and augmenting models in general, and spatial interaction ones in particular. These three routes are aligned around the following principles:

1. Use better approximations to model your dependent variable.
2. Recognize the structure of your data.
3. Get better predictors.

- **Use better approximations to model your dependent variable**

Standard OLS regression assumes that the error term and, since the predictors are deterministic, the dependent variable are distributed following a normal (gaussian) distribution. This is usually a good approximation for several phenomena of interest, but maybe not the best one for trips along routes: for one, we know trips cannot be negative, which the normal distribution does not account for; more subtly, their distribution is not really symmetric but skewed with a very long tail on the right. This is common in variables that represent counts and that is why usually it is more appropriate to fit a model that relies on a distribution different from the normal.

One of the most common distributions for this cases is the Poisson, which can be incorporated through a general linear model (or GLM). The underlying assumption here is that instead of $T_{ij} \sim N(\mu_{ij}, \sigma)$, our model now follows:

$$T_{ij} \sim Poisson(\exp^{X_i \beta})$$

---

3The $R^2$ of our model is around 2%.
4which they are not really, in light of the comparison between the black and red lines.
5These principles are general and can be applied to pretty much any modeling exercise you run into. The specific approaches we take in this note relate to spatial interaction models.
6For an illustration of this, consider the amount of probability mass to the left of zero in the predictive checks above.
As usual, such a model is easy to run in R:

```r
m2 <- glm(
  'trips15 ~ straight_dist + total_up + total_down',
  data=db_std,
  family=poisson,
)
```

Now let’s see how much better, if any, this approach is. To get a quick overview, we can simply plot the point predictions:

```r
plot(
  density(m2$fitted.values),
  xlim=c(-100, max(db_std$trips15)),
  ylim=c(0, max(c(
    max(density(m2$fitted.values)$y),
    max(density(db_std$trips15)$y)
  ))),
  col='black',
  main='')
lines(
  density(db_std$trips15),
  col='red',
  main='')
legend(
  'topright',
  c('Predicted', 'Actual'),
  col=c('black', 'red'),
  lwd=1)
title(main="Predictive check, point estimates - Poisson model")
```
To incorporate uncertainty to these predictions, we need to tweak our `generate_draw` function so it accommodates the fact that our model is not linear anymore.

```r
generate_draw_poi <- function(m) {
  # Set up predictors matrix
  x <- model.matrix(m)
  # Obtain draws of parameters (inferential uncertainty)
  sim.bs <- sim(m, 1)
  # Predicted value
  xb <- x %*% sim.bs@coef[1,]
  # Transform using the link function
  mu <- exp(xb)
  # Obtain a random realization
  y_hat <- rpois(n=length(mu), lambda=mu)
  return(y_hat)
}
```

And then we can examine both point predictions an uncertainty around them:

```r
plot(
  density(m2$fitted.values),
  xlim=c(-100, max(db.std$trips15)),
  ylim=c(0, max(c(
    max(density(m2$fitted.values)$y),
```
max(density(db_std$trips15)$y)
)
)
}
col='white',
main=''
)
# Loop for realizations
for(i in 1:250){
tmp_y <- generate_draw.poi(m2)
lines(
density(tmp_y),
col='grey',
lwd=0.1
)
}
#
lines(
density(m2$fitted.values),
col='black',
main=''
)
lines(
density(db_std$trips15),
col='red',
main=''
)
legend(
'topleft',
c('Predicted', 'Actual', 'Simulated (n=250)'),
col=c('black', 'red', 'grey'),
lwd=1
)
title(main="Predictive check - Poisson model")
Voila! Although the curve is still a bit off, centered too much to the right of the actual data, our predictive simulation leaves the fitted values right in the middle. This speaks to a better fit of the model to the actual distribution of the original data follow.

- **Recognize the structure of your data**

So far, we've treated our dataset as if it was flat (i.e. comprise of fully independent realizations) when in fact it is not. Most crucially, our baseline model does not account for the fact that every observation in the dataset pertains to a trip between two stations. This means that all the trips from or to the same station probably share elements which likely help explain how many trips are undertaken between stations. For example, think of trips to and from a station located in the famous Embarcadero, a popular tourist spot. Every route to and from there probably has more trips due to the popularity of the area and we are currently not acknowledging it in the model.

A simple way to incorporate these effects into the model is through origin and destination fixed effects. This approach shares elements with both spatial fixed effects and multilevel modeling and essentially consists of including a binary variable for every origin and destination station. In mathematical notation, this equates to:

\[ T_{ij} = X_{ij} \beta + \delta_i + \delta_j + \epsilon_{ij} \]
where $\delta_i$ and $\delta_j$ are origin and destination station fixed effects\(^7\), and the rest is as above. This strategy accounts for all the unobserved heterogeneity associated with the location of the station. Technically speaking, we simply need to introduce \texttt{orig} and \texttt{dest} in the the model:

```r
m3 <- glm(
  'trips15 ~ straight_dist + total_up + total_down + orig + dest',
  data=db_std,
  family=poisson
)
```

And with our new model, we can have a look at how well it does at predicting the overall number of trips\(^8\):

```r
plot(
  density(m3$fitted.values),
  xlim=c(-100, max(db_std$trips15)),
  ylim=c(0, max(c(
    max(density(m3$fitted.values)$y),
    max(density(db_std$trips15)$y)
  )
  ),
  col='black',
  main='',
)
lines(
  density(db_std$trips15),
  col='red',
  main='',
)
legend(
  'topright',
  c('Predicted', 'Actual'),
  col=c('black', 'red'),
  lwd=1
)
```

\(^7\)In this session, $\delta_i$ and $\delta_j$ are estimated as independent variables so their estimates are similar to interpret to those in $\beta$. An alternative approach could be to model them as random effects in a multilevel framework.

\(^8\)Although, theoretically, we could also include simulations of the model in the plot to get a better sense of the uncertainty behind our model, in practice this seems troublesome. The problems most likely arise from the fact that many of the origin and destination binary variable coefficients are estimated with a great deal of uncertainty. This causes some of the simulation to generate extreme values that, when passed through the exponential term of the Poisson link function, cause problems. If anything, this is testimony of how a simple fixed effect model can sometimes lack accuracy and generate very uncertain estimates. A potential extension to work around these problems could be to fit a multilevel model with two specific levels beyond the trip-level: one for origin and another one for destination stations.
That looks significantly better, doesn’t it? In fact, our model now better accounts for the long tail where a few routes take a lot of trips. This is likely because the distribution of trips is far from random across stations and our origin and destination fixed effects do a decent job at accounting for that structure. However our model is still notably underpredicting less popular routes and over-predicting routes with above average number of trips. Maybe we should think about moving beyond a simple linear model.

- **Get better predictors**

The final extension is, in principle, always available but, in practice, it can be tricky to implement. The core idea is that your baseline model might not have the best measurement of the phenomena you want to account for. In our example, we can think of the distance between stations. So far, we have been including the distance measured “as the crow flies” between stations. Although in some cases this is a good approximation (particularly when distances are long and likely route taken is as close to straight as possible), in some cases like ours, where the street layout and the presence of elevation probably matter more than the actual final distance pedalled, this is not necessarily a safe assumption.

As an example of this approach, we can replace the straight distance measurements for more refined ones based on the Google Maps API routes. This is very easy as all we need to do (once the distances have been calculated!) is to swap `straight_dist` for `street_dist`:
m4 <- glm(
  'trips15 ~ street_dist + total_up + total_down + orig + dest',
  data=db_std,
  family=poisson
)

And we can similarly get a sense of our predictive fitting with:

plot(
  density(m4$fitted.values),
  xlim=c(-100, max(db_std$trips15)),
  ylim=c(0, max(c(
    max(density(m4$fitted.values)$y),
    max(density(db_std$trips15)$y)
  )))
),
  col='black',
  main=''
)
lines(
  density(db_std$trips15),
  col='red',
  main=''
)
legend(
  'topright',
  c('Predicted', 'Actual'),
  col=c('black', 'red'),
  lwd=1
)
title(main="Predictive check - Orig/dest FE Poisson model")
Hard to tell any noticeable difference, right? To see if there is any, we can have a look at the estimates obtained:

```
summary(m4)$coefficients['street_dist', ]
```

```
## Estimate Std. Error  z value Pr(>|z|)
## -9.961619e-02 2.688731e-03 -3.704952e+01 1.828096e-300
```

And compare this to that of the straight distances in the previous model:

```
summary(m3)$coefficients['straight_dist', ]
```

```
## Estimate Std. Error  z value Pr(>|z|)
## -7.820014e-02 2.683052e-03 -2.914596e+01 9.399407e-187
```

As we can see, the differences exist but are not massive. Let’s use this example to learn how to interpret coefficients in a Poisson model. Effectively, these estimates can be understood as multiplicative effects. Since our model fits

\[ T_{ij} \sim \text{Poisson}(\exp^{X_{ij} \beta}) \]

we need to transform \( \beta \) through an exponential in order to get a sense of the effect of distance on the number of trips. This means that for the street distance, our original estimate is \( \beta_{\text{street}} = -0.0996 \), but this needs to be translated through the exponential into \( e^{-0.0996} = 0.906 \). In other words, since distance is expressed

---

9See section 6.2 of Gelman and Hill (2006a) for a similar treatment of these.
in standard deviations\textsuperscript{10}, we can expect a 10\% decrease in the number of trips for an increase of one standard deviation (about 1Km) in the distance between the stations. This can be compared with $e^{-0.0782} = 0.925$ for the straight distances, or a reduction of about 8\% the number of trips for every increase of a standard deviation (about 720m).

5.5 Predicting flows

So far we have put all of our modeling efforts in understanding the model we fit and improving such model so it fits our data as closely as possible. This is essential in any modelling exercise but should be far from a stopping point. Once we’re confident our model is a decent representation of the data generating process, we can start exploiting it. In this section, we will cover one specific case that showcases how a fitted model can help: out-of-sample forecasts.

It is August 2015, and you have just started working as a data scientist for the bikeshare company that runs the San Francisco system. You join them as they’re planning for the next academic year and, in order to plan their operations (re-allocating vans, station maintenance, etc.), they need to get a sense of how many people are going to be pedalling across the city and, crucially, where they are going to be pedalling through. What can you do to help them?

The easiest approach is to say “well, a good guess for how many people will be going between two given stations this coming year is how many went through last year, isn’t it?” This is one prediction approach. However, you could see how, even if the same process governs over both datasets (2015 and 2016), each year will probably have some idiosyncracies and thus looking too closely into one year might not give the best possible answer for the next one. Ideally, you want a good stylized synthesis that captures the bits that stay constant over time and thus can be applied in the future and that ignores those aspects that are too particular to a given point in time. That is the rationale behind using a fitted model to obtain predictions.

However good any theory though, the truth is in the pudding. So, to see if a modeling approach is better at producing forecasts than just using the counts from last year, we can put them to a test. The way this is done when evaluating the predictive performance of a model (as this is called in the literature) relies on two basic steps: a) obtain predictions from a given model and b) compare those to the actual values (in our case, with the counts for 2016 in trips\textsubscript{16}) and get a sense of “how off” they are. We have essentially covered a) above; for b), there are several measures to use. We will use one of the most common ones, the root mean squared error (RMSE), which roughly gives a sense of the average difference between a predicted vector and the real deal:

\textsuperscript{10}Remember the transformation at the very beginning.
5.5. PREDICTING FLOWS

\[ RMSE = \sqrt{\sum_{ij} (\hat{T}_{ij} - T_{ij})^2} \]

where \( \hat{T}_{ij} \) is the predicted amount of trips between stations \( i \) and \( j \). RMSE is straightforward in R and, since we will use it a couple of times, let’s write a short function to make our lives easier:

```r
rmse <- function(t, p) {
  se <- (t - p)^2
  mse <- mean(se)
  rmse <- sqrt(mse)
  return(rmse)
}
```

where \( t \) stands for the vector of true values, and \( p \) is the vector of predictions. Let’s give it a spin to make sure it works:

```r
rmse_m4 <- rmse(db_std$trips16, m4$fitted.values)
rmse_m4
## [1] 256.2197
```

That means that, on average, predictions in our best model \( m4 \) are 256 trips off. Is this good? Bad? Worse? It’s hard to say but, being practical, what we can say is whether this better than our alternative. Let us have a look at the RMSE of the other models as well as that of simply plugging in last year’s counts:\(^{11}\)

```r
rmse <- data.frame(
  model=c('OLS',
         'Poisson',
         'Poisson + FE',
         'Poisson + FE + street dist.',
         'Trips-2015'),
  RMSE=c(rmse(db_std$trips16, m1$fitted.values),
         rmse(db_std$trips16, m2$fitted.values),
         rmse(db_std$trips16, m3$fitted.values),
         rmse(db_std$trips16, m4$fitted.values),
         rmse(db_std$trips16, db_std$trips15))
)
```

\(^{11}\)EXERCISE: can you create a single plot that displays the distribution of the predicted values of the five different ways to predict trips in 2016 and the actual counts of trips?
## model RMSE

<table>
<thead>
<tr>
<th></th>
<th>model</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OLS</td>
<td>323.6135</td>
</tr>
<tr>
<td>2</td>
<td>Poisson</td>
<td>320.8962</td>
</tr>
<tr>
<td>3</td>
<td>Poisson + FE</td>
<td>254.4468</td>
</tr>
<tr>
<td>4</td>
<td>Poisson + FE + street dist.</td>
<td>256.2197</td>
</tr>
<tr>
<td>5</td>
<td>Trips-2015</td>
<td>131.0228</td>
</tr>
</tbody>
</table>

The table is both encouraging and disheartening at the same time. On the one hand, all the modeling techniques covered above behave as we would expect: the baseline model displays the worst predicting power of all, and every improvement (except the street distances!) results in notable decreases of the RMSE. This is good news. However, on the other hand, all of our modelling efforts fall short of given a better guess than simply using the previous year’s counts. *Why? Does this mean that we should not pay attention to modeling and inference?* Not really. Generally speaking, a model is as good at predicting as it is able to mimic the underlying process that gave rise to the data in the first place. The results above point to a case where our model is not picking up all the factors that determine the amount of trips undertaken in a give route. This could be improved by enriching the model with more/better predictors, as we have seen above. Also, the example above seems to point to a case where those idiosyncracies in 2015 that the model does not pick up seem to be at work in 2016 as well. This is great news for our prediction efforts this time, but we have no idea why this is the case and, for all that matters, it could change the coming year. Besides the elegant quantification of uncertainty, the true advantage of a modeling approach in this context is that, if well fit, it is able to pick up the fundamentals that apply over and over. This means that, if next year we’re not as lucky as this one and previous counts are not good predictors but the variables we used in our model continue to have a role in determining the outcome, the data scientist should be luckier and hit a better prediction.

### 5.6 Questions

We will be using again the Madrid AirBnb dataset:

```r
mad_abb <- readOGR('./data/assignment_1_madrid/madrid_abb.gpkg')
```

```
## OGR data source with driver: GPKG
## Source: "./home/jovyan/work/data/assignment_1_madrid/madrid_abb.gpkg", layer: "madrid_abb"
## with 18399 features
## It has 16 fields
## Integer64 fields read as strings: accommodates
```

The columns to use here are:

- `price_usd`: price expressed in USD
- `log1p_price_usd`: logarithm of the price expressed in USD
- `accommodates`: number of people the property accommodates
5.6. QUESTIONS

- **bathrooms**: number of bathrooms the property includes
- **bedrooms**: number of bedrooms the property includes
- **beds**: number of beds the property includes

With these data at hand, accomplish the following challenges:

1. Set up a baseline regression model where you explain the price of a property as a function of its characteristics:

   \[ P_i = \alpha + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + \epsilon_i \]

2. Fit a parallel model that uses the log of price as dependent variable:

   \[ \log(P_i) = \alpha + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + \epsilon_i \]

3. Perform a predictive check analysis of both models, discussing how they compare, which one you would prefer, and why
Chapter 6

Spatial Econometrics

This chapter is based on the following references, which are good follow-up's on the topic:

- Chapter 11 of the GDS Book, by Rey et al. (ming).
- Session III of Arribas-Bel (2014). Check the “Related readings” section on the session page for more in-depth discussions.
- Anselin (2007), freely available to download [pdf].
- The second part of this tutorial assumes you have reviewed Block E of Arribas-Bel (2019). [html]

6.1 Dependencies

We will rely on the following libraries in this section, all of them included in the book list:

```r
# Layout
library(tufte)
# For pretty table
library(knitr)
# For string parsing
library(stringr)
# Spatial Data management
library(rgdal)
# Pretty graphics
library(ggplot2)
# Pretty maps
library(ggmap)
# Various GIS utilities
library(GISTools)
# For all your interpolation needs
```
library(gstat)
# For data manipulation
library(dplyr)
# Spatial regression
library(spdep)

Before we start any analysis, let us set the path to the directory where we are working. We can easily do that with setwd(). Please replace in the following line the path to the folder where you have placed this file - and where the house_transactions folder with the data lives.

setwd('.')

6.2 Data

To explore ideas in spatial regression, we will the set of Airbnb properties for San Diego (US), borrowed from the “Geographic Data Science with Python” book (see here for more info on the dataset source). This covers the point location of properties advertised on the Airbnb website in the San Diego region.

Let us load it up first of all:

db <- st_read('data/abb_sd/regression_db.geojson')

## Reading layer `regression_db' from data source `/home/jovyan/work/data/abb_sd/regression_db.geojson' using driver `GeoJSON'
## Simple feature collection with 6110 features and 19 fields
## geometry type: POINT
## dimension: XY
## geographic CRS: WGS 84

The table contains the following variables:

names(db)

## [1] "accommodates"  "bathrooms"   "bedrooms"
## [4] "beds"          "neighborhood" "pool"
## [7] "d2balboa"      "coastal"      "price"
## [10] "log_price"     "id"           "pg_Apartment"
## [13] "pg_Condominium" "pg_House"     "pg_Other"
## [16] "pg_Townhouse"  "rt_Entire_home.apt" "rt_Private_room"
## [19] "rt_Shared_room" "geometry"

For most of this chapter, we will be exploring determinants and strategies for modelling the price of a property advertised in Airbnb. To get a first taste of what this means, we can create a plot of prices within the area of San Diego:
6.3 Non-spatial regression, a refresh

Before we discuss how to explicitly include space into the linear regression framework, let us show how basic regression can be carried out in R, and how you can interpret the results. By no means is this a formal and complete introduction to regression so, if that is what you are looking for, the first part of Gelman and Hill (2006a), in particular chapters 3 and 4, are excellent places to check out.

The core idea of linear regression is to explain the variation in a given (dependent) variable as a linear function of a series of other (explanatory) variables. For example, in our case, we may want to express/explain the price of a property advertised on AirBnb as a function of some of its characteristics, such as the number of people it accommodates, and how many bathrooms, bedrooms and beds it features. At the individual level, we can express this as:

$$\log(P_i) = \alpha + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + \epsilon_i$$

where $P_i$ is the price of house $i$, $Acc_i$, $Bath_i$, $Bedr_i$ and $Beds_i$ are the count
of people it accommodates, bathrooms, bedrooms and beds that house \( i \) has, respectively. The parameters \( \beta_{1,2,3,4} \) give us information about in which way and to what extent each variable is related to the price, and \( \alpha \), the constant term, is the average house price when all the other variables are zero. The term \( \epsilon_i \) is usually referred to as the “error” and captures elements that influence the price of a house but are not accounted for explicitly. We can also express this relation in matrix form, excluding subindices for \( i \) as:

\[
\log(P) = \alpha + \beta_1 Acc + \beta_2 Bath + \beta_3 Bedr + \beta_4 Beds + \epsilon
\]

where each term can be interpreted in terms of vectors instead of scalars (wit the exception of the parameters (\( \alpha, \beta_{1,2,3,4} \), which are scalars). Note we are using the logarithm of the price, since this allows us to interpret the coefficients as roughly the percentual change induced by a unit increase in the explanatory variable of the estimate.

Remember a regression can be seen as a multivariate extension of bivariate correlations. Indeed, one way to interpret the \( \beta_k \) coefficients in the equation above is as the degree of correlation between the explanatory variable \( k \) and the dependent variable, *keeping all the other explanatory variables constant*. When you calculate simple bivariate correlations, the coefficient of a variable is picking up the correlation between the variables, but it is also subsuming into it variation associated with other correlated variables –also called confounding factors\(^1\). Regression allows you to isolate the distinct effect that a single variable has on the dependent one, once we *control* for those other variables.

Practically speaking, running linear regressions in \( R \) is straightforward. For example, to fit the model specified in the equation above, we only need one line of code:

```r
m1 <- lm('log_price ~ accommodates + bathrooms + bedrooms + beds', db)
```

We use the command *lm*, for linear model, and specify the equation we want to fit using a string that relates the dependent variable (the log of the price, \( \text{log\_price} \)) with a set of explanatory ones (\( \text{accommodates, bathrooms, bedrooms, beds} \)) by using a tilde ~ that is akin to the = symbol in the mathematical equation above. Since we are using names of variables that are stored in a table, we need to pass the table object (\( \text{db} \)) as well.

In order to inspect the results of the model, the quickest way is to call *summary*:

---

\(^1\) **EXAMPLE** Assume that new houses tend to be built more often in areas with low deprivation. If that is the case, then \( NEW \) and \( IMD \) will be correlated with each other (as well as with the price of a house, as we are hypothesizing in this case). If we calculate a simple correlation between \( P \) and \( IMD \), the coefficient will represent the degree of association between both variables, but it will also include some of the association between \( IMD \) and \( NEW \). That is, part of the obtained correlation coefficient will be due not to the fact that higher prices tend to be found in areas with low IMD, but to the fact that new houses tend to be more expensive. This is because (in this example) new houses tend to be built in areas with low deprivation and simple bivariate correlation cannot account for that.
6.4. SPATIAL REGRESSION: A (VERY) FIRST DIP

summary(m1)

## Call:
## lm(formula = "log_price ~ accommodates + bathrooms + bedrooms + beds",
##     data = db)
##
## Residuals:
##    Min     1Q   Median     3Q    Max
## -2.8486 -0.3234  0.0095  0.3023  3.3975
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.018133   0.013947 288.10  < 2e-16 ***
## accommodates 0.176851   0.005323  33.23  < 2e-16 ***
## bathrooms  0.150981   0.012526  12.05  < 2e-16 ***
## bedrooms   0.111700   0.012537   8.91  < 2e-16 ***
## beds  -0.076974   0.007927  -9.71  < 2e-16 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5366 on 6105 degrees of freedom
## Multiple R-squared: 0.5583, Adjusted R-squared: 0.558
## F-statistic: 1929 on 4 and 6105 DF,  p-value: < 2.2e-16

A full detailed explanation of the output is beyond the scope of the chapter, but we will highlight the relevant bits for our main purpose. This is concentrated on the Coefficients section, which gives us the estimates for the $\beta_k$ coefficients in our model. These estimates are the raw equivalent of the correlation coefficient between each explanatory variable and the dependent one, once the “polluting” effect of the other variables included in the model has been accounted for. Results are as expected for the most part: houses tend to be significantly more expensive if they accommodate more people (an extra person increases the price by 17.7%, approximately), have more bathrooms (15.1%), or bedrooms (11.2%). Perhaps counter intuitively, an extra bed available seems to decrease the price by about -7.7%. However, keep in mind that this is the case, *everything else equal*. Hence, more beds per room and bathroom (ie. a more crowded house), is a bit cheaper.

6.4 Spatial regression: a (very) first dip

Spatial regression is about *explicitly* introducing space or geographical context into the statistical framework of a regression. Conceptually, we want to intro-
duce space into our model whenever we think it plays an important role in the process we are interested in, or when space can act as a reasonable proxy for other factors we cannot but should include in our model. As an example of the former, we can imagine how houses at the seafront are probably more expensive than those in the second row, given their better views. To illustrate the latter, we can think of how the character of a neighborhood is important in determining the price of a house; however, it is very hard to identify and quantify “character” per se, although it might be easier to get at its spatial variation, hence a case of space as a proxy.

Spatial regression is a large field of development in the econometrics and statistics literatures. In this brief introduction, we will consider two related but very different processes that give rise to spatial effects: spatial heterogeneity and spatial dependence. For more rigorous treatments of the topics introduced here, the reader is referred to Anselin (2003), Anselin and Rey (2014), and Gibbons et al. (2014).

6.5 Spatial heterogeneity

Spatial heterogeneity (SH) arises when we cannot safely assume the process we are studying operates under the same “rules” throughout the geography of interest. In other words, we can observe SH when there are effects on the outcome variable that are intrinsically linked to specific locations. A good example of this is the case of seafront houses above: we are trying to model the price of a house and, the fact some houses are located under certain conditions (i.e. by the sea), makes their price behave differently. This somewhat abstract concept of SH can be made operational in a model in several ways. We will explore the following two: spatial fixed-effects (FE); and spatial regimes, which is a generalization of FE.

Spatial FE

Let us consider the house price example from the previous section to introduce a more general illustration that relates to the second motivation for spatial effects (“space as a proxy”). Given we are only including two explanatory variables in the model, it is likely we are missing some important factors that play a role at determining the price at which a house is sold. Some of them, however, are likely to vary systematically over space (e.g. different neighborhood characteristics). If that is the case, we can control for those unobserved factors by using traditional dummy variables but basing their creation on a spatial rule. For example, let us include a binary variable for every neighbourhood, as provided by AirBnB, indicating whether a given house is located within such area (1) or not (0). Neighbourhood membership is expressed on the neighborhood column:

db %>%
  ggplot(aes(color = neighborhood)) +
  geom_sf() +

# More code here...
Mathematically, we are now fitting the following equation:

$$\log(P_i) = \alpha_r + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + \epsilon_i$$

where the main difference is that we are now allowing the constant term, $\alpha$, to vary by neighbourhood $r$, $\alpha_r$.

Programmatically, we can fit this model with `lm`:

```
# Include `-1` to eliminate the constant term and include a dummy for every area
m2 <- lm(
  "log_price ~ neighborhood + accommodates + bathrooms + bedrooms + beds - 1",
  data = db
)
summary(m2)
```

## Call:
```
lm(formula = "log_price ~ neighborhood + accommodates + bathrooms + bedrooms + beds - 1",
    data = db)
```

## Residuals:
```
  Min   1Q Median   3Q  Max
-2.4549 -0.2920 -0.0203 0.2741 3.5323
```
## Coefficients:

| Neighborhood                  | Estimate | Std. Error | t value | Pr(>|t|) |
|------------------------------|----------|------------|---------|----------|
| neighborhoodBalboa Park      | 3.994775 | 0.036539   | 109.33  | <2e-16 *** |
| neighborhoodBay Ho           | 3.780025 | 0.086081   | 43.91   | <2e-16 *** |
| neighborhoodBay Park         | 3.941847 | 0.055788   | 70.66   | <2e-16 *** |
| neighborhoodCarmel Valley    | 4.034052 | 0.062811   | 64.23   | <2e-16 *** |
| neighborhoodCity Heights West| 3.698788 | 0.051438   | 71.12   | <2e-16 *** |
| neighborhoodClairemont Mesa | 3.658339 | 0.054979   | 67.66   | <2e-16 *** |
| neighborhoodCollege Area     | 3.649859 | 0.059438   | 67.12   | <2e-16 *** |
| neighborhoodCore             | 4.433447 | 0.058864   | 75.32   | <2e-16 *** |
| neighborhoodCortez Hill      | 4.294790 | 0.057648   | 74.50   | <2e-16 *** |
| neighborhoodDel Mar Heights  | 4.300659 | 0.060912   | 70.61   | <2e-16 *** |
| neighborhoodEast Village     | 4.241146 | 0.032019   | 132.46  | <2e-16 *** |
| neighborhoodGaslamp Quarter  | 4.473863 | 0.052493   | 85.23   | <2e-16 *** |
| neighborhoodGrant Hill       | 4.001481 | 0.058825   | 68.02   | <2e-16 *** |
| neighborhoodGrantville       | 3.664989 | 0.080168   | 45.72   | <2e-16 *** |
| neighborhoodKensington       | 4.073520 | 0.087322   | 46.65   | <2e-16 *** |
| neighborhoodLa Jolla         | 4.400145 | 0.026772   | 164.36  | <2e-16 *** |
| neighborhoodLa Jolla Village | 4.066151 | 0.087263   | 46.60   | <2e-16 *** |
| neighborhoodLinda Vista      | 3.817940 | 0.063128   | 60.48   | <2e-16 *** |
| neighborhoodLittle Italy     | 4.390651 | 0.052433   | 83.74   | <2e-16 *** |
| neighborhoodLoma Portal      | 4.034473 | 0.036173   | 111.53  | <2e-16 *** |
| neighborhoodMarina           | 4.046133 | 0.052178   | 77.55   | <2e-16 *** |
| neighborhoodMidtown          | 4.032038 | 0.030280   | 133.16  | <2e-16 *** |
| neighborhoodMidtown District | 4.356943 | 0.071766   | 60.72   | <2e-16 *** |
| neighborhoodMira Mesa        | 3.570523 | 0.061543   | 58.02   | <2e-16 *** |
| neighborhoodMission Bay      | 4.251309 | 0.023318   | 182.32  | <2e-16 *** |
| neighborhoodMission Valley  | 4.012410 | 0.083766   | 47.90   | <2e-16 *** |
| neighborhoodMoreno Mission   | 4.028288 | 0.063342   | 63.59   | <2e-16 *** |
| neighborhoodNormal Heights   | 3.791895 | 0.054730   | 69.28   | <2e-16 *** |
| neighborhoodNorth Clairemont| 3.498107 | 0.076432   | 45.77   | <2e-16 *** |
| neighborhoodNorth Hills      | 3.959403 | 0.026823   | 147.61  | <2e-16 *** |
| neighborhoodNorthwest        | 3.810201 | 0.071586   | 48.75   | <2e-16 *** |
| neighborhoodOcean Beach      | 4.152695 | 0.032365   | 128.36  | <2e-16 *** |
| neighborhoodOld Town         | 4.127737 | 0.046523   | 88.72   | <2e-16 *** |
| neighborhoodOtay Ranch       | 3.722902 | 0.091633   | 40.63   | <2e-16 *** |
| neighborhoodPacific Beach    | 4.116749 | 0.022711   | 181.27  | <2e-16 *** |
| neighborhoodPark West        | 4.216829 | 0.050370   | 83.72   | <2e-16 *** |
| neighborhoodRancho Bernadino | 3.873962 | 0.080780   | 47.96   | <2e-16 *** |
| neighborhoodRancho Penasquitos| 3.772037| 0.068808   | 54.82   | <2e-16 *** |
| neighborhoodRoseville        | 4.070468 | 0.065299   | 62.34   | <2e-16 *** |
| neighborhoodSan Carlos       | 3.935042 | 0.093205   | 42.22   | <2e-16 *** |
| neighborhoodScripps Ranch    | 3.641239 | 0.085190   | 42.74   | <2e-16 *** |
| neighborhoodSerra Mesa       | 3.912127 | 0.066630   | 58.71   | <2e-16 *** |
| neighborhoodSouth Park       | 3.987019 | 0.060141   | 66.30   | <2e-16 *** |
Econometrically speaking, what the postcode FE we have introduced imply is that, instead of comparing all house prices across San Diego as equal, we only derive variation from within each postcode. In our particular case, estimating spatial FE in our particular example also gives you an indirect measure of area desirability: since they are simple dummies in a regression explaining the price of a house, their estimate tells us about how much people are willing to pay to live in a given area. We can visualise this “geography of desirability” by plotting the estimates of each fixed effect on a map:

```r
# Extract neighborhood names from coefficients
nei.names <- m2$coefficients %>%
  as.data.frame() %>%
  row.names() %>%
  str_replace("neighborhood", "")

# Set up as Data Frame
nei.fes <- data.frame(
  coef = m2$coefficients,
  nei = nei.names,
  row.names = nei.names
)%>%
  right_join(
    db, by = c("nei" = "neighborhood")
  )

# Plot
nei.fes %>%
  st_as_sf() %>%
  ggplot(aes(color = coef)) +
  geom_sf() +
  scale_color_viridis_c() +
  theme_void()
```
We can see how neighborhoods in the left (west) tend to have higher prices. What we can’t see, but it is represented there if you are familiar with the geography of San Diego, is that the city is bounded by the Pacific ocean on the left, suggesting neighbourhoods by the beach tend to be more expensive.

Remember that the interpretation of a $\beta_k$ coefficient is the effect of variable $k$, given all the other explanatory variables included remain constant. By including a single variable for each area, we are effectively forcing the model to compare as equal only house prices that share the same value for each variable; in other words, only houses located within the same area. Introducing FE affords you a higher degree of isolation of the effects of the variables you introduce in your model because you can control for unobserved effects that align spatially with the distribution of the FE you introduce (by neighbourhood, in our case).

**Spatial regimes**

At the core of estimating spatial FE's is the idea that, instead of assuming the dependent variable behaves uniformly over space, there are systematic effects following a geographical pattern that affect its behaviour. In other words, spatial FE's introduce econometrically the notion of spatial heterogeneity. They do this in the simplest possible form: by allowing the constant term to vary geographically. The other elements of the regression are left untouched and hence apply uniformly across space. The idea of spatial regimes (SRs) is to generalize the spatial FE approach to allow not only the constant term to vary but also any other explanatory variable. This implies that the equation we will be estimating is:

$$\log(P_i) = \alpha_r + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + \epsilon_i$$
where we are not only allowing the constant term to vary by region ($\alpha_r$), but also every other parameter ($\beta_{kr}$).

Also, given we are going to allow every coefficient to vary by regime, we will need to explicitly set a constant term that we can allow to vary:

```r
db$one <- 1
```

Then, the estimation leverages the capabilities in model description of R formulas:

```r
# ':.' notation implies interaction variables
m3 <- lm(
  'log_price ~ 0 + (accommodates + bathrooms + bedrooms + beds):(neighborhood)',
  db
)
summary(m3)
```

```r
## Call:
## lm(formula = "log_price ~ 0 + (accommodates + bathrooms + bedrooms + beds):(neighborhood)",
##     data = db)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -10.4790 -0.0096  1.0931  1.7599  6.1073
##
## Coefficients:
##             Estimate  Std. Error t value
## accommodates:neighborhoodBalboa Park  0.063528  0.093237   0.681
## accommodates:neighborhoodBay Ho -0.259615  0.335007  -0.775
## accommodates:neighborhoodBay Park -0.355401  0.232720  -1.527
## accommodates:neighborhoodCarmel Valley  0.129786  0.187193   0.693
## accommodates:neighborhoodCity Heights West  0.447371  0.231998   1.928
## accommodates:neighborhoodClairemont Mesa  0.711353  0.177821   4.000
## accommodates:neighborhoodCollege Area -0.346152  0.188071  -1.841
## accommodates:neighborhoodCore  0.125864  0.148417   0.848
## accommodates:neighborhoodCortez Hill  0.715958  0.126562   5.657
## accommodates:neighborhoodDel Mar Heights  0.829195  0.214067   3.874
## accommodates:neighborhoodEast Village  0.214642  0.077394   2.773
## accommodates:neighborhoodGaslamp Quarter  0.451443  0.197637   2.284
## accommodates:neighborhoodGrant Hill 1.135176  0.167771   6.766
## accommodates:neighborhoodGrantville  0.300907  0.280369   1.073
## accommodates:neighborhoodKensington  0.668742  0.450243   1.485
## accommodates:neighborhoodLa Jolla  0.520882  0.058887   9.030
## accommodates:neighborhoodLa Jolla Village  0.566452  0.413185   1.371
## accommodates:neighborhoodLinda Vista  0.523975  0.282219   1.857
## accommodates:neighborhoodLittle Italy  0.603908  0.121899   4.954
```
<table>
<thead>
<tr>
<th>Neighborhood</th>
<th>Value1</th>
<th>Value2</th>
<th>Value3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loma Portal</td>
<td>0.487743</td>
<td>0.127870</td>
<td>3.814</td>
</tr>
<tr>
<td>Marina</td>
<td>0.431384</td>
<td>0.172628</td>
<td>2.499</td>
</tr>
<tr>
<td>Midtown</td>
<td>0.618058</td>
<td>0.087992</td>
<td>7.024</td>
</tr>
<tr>
<td>Midtown District</td>
<td>0.430398</td>
<td>0.191682</td>
<td>2.245</td>
</tr>
<tr>
<td>Mira Mesa</td>
<td>-0.018199</td>
<td>0.310167</td>
<td>-0.059</td>
</tr>
<tr>
<td>Mission Bay</td>
<td>0.440951</td>
<td>0.049454</td>
<td>8.916</td>
</tr>
<tr>
<td>Mission Valley</td>
<td>0.144530</td>
<td>0.507925</td>
<td>0.285</td>
</tr>
<tr>
<td>Normal Heights</td>
<td>0.413682</td>
<td>0.198584</td>
<td>2.083</td>
</tr>
<tr>
<td>North Clairemont</td>
<td>-0.242723</td>
<td>0.307090</td>
<td>-0.790</td>
</tr>
<tr>
<td>North Hills</td>
<td>0.262840</td>
<td>0.083258</td>
<td>3.157</td>
</tr>
<tr>
<td>Northwest</td>
<td>-0.229157</td>
<td>0.255656</td>
<td>-0.896</td>
</tr>
<tr>
<td>Ocean Beach</td>
<td>0.754771</td>
<td>0.079097</td>
<td>9.542</td>
</tr>
<tr>
<td>Old Town</td>
<td>0.177176</td>
<td>0.159714</td>
<td>1.109</td>
</tr>
<tr>
<td>Day ranch</td>
<td>-0.333536</td>
<td>0.309654</td>
<td>-1.078</td>
</tr>
<tr>
<td>Pacific Beach</td>
<td>0.345475</td>
<td>0.057599</td>
<td>5.998</td>
</tr>
<tr>
<td>Park West</td>
<td>0.909020</td>
<td>0.156013</td>
<td>5.827</td>
</tr>
<tr>
<td>Rancho Bernadino</td>
<td>-0.118939</td>
<td>0.256750</td>
<td>-0.463</td>
</tr>
<tr>
<td>Rancho Penasquitos</td>
<td>0.121845</td>
<td>0.228456</td>
<td>0.533</td>
</tr>
<tr>
<td>Roseville</td>
<td>0.316929</td>
<td>0.226110</td>
<td>1.402</td>
</tr>
<tr>
<td>San Carlos</td>
<td>0.191248</td>
<td>0.318706</td>
<td>0.600</td>
</tr>
<tr>
<td>Scripps Ranch</td>
<td>0.347638</td>
<td>0.127239</td>
<td>2.732</td>
</tr>
<tr>
<td>Serra Mesa</td>
<td>0.495491</td>
<td>0.282281</td>
<td>1.755</td>
</tr>
<tr>
<td>South Park</td>
<td>0.334378</td>
<td>0.256708</td>
<td>1.303</td>
</tr>
<tr>
<td>University City</td>
<td>0.107605</td>
<td>0.113883</td>
<td>0.945</td>
</tr>
<tr>
<td>West University Heights</td>
<td>0.190215</td>
<td>0.212040</td>
<td>0.897</td>
</tr>
<tr>
<td>Balboa Park</td>
<td>2.275321</td>
<td>0.225032</td>
<td>10.111</td>
</tr>
<tr>
<td>Bay Ho</td>
<td>3.312231</td>
<td>0.530568</td>
<td>6.243</td>
</tr>
<tr>
<td>Bay Park</td>
<td>2.231649</td>
<td>0.365655</td>
<td>6.103</td>
</tr>
<tr>
<td>Carmel Valley</td>
<td>1.191058</td>
<td>0.224138</td>
<td>5.314</td>
</tr>
<tr>
<td>City Heights West</td>
<td>2.517235</td>
<td>0.550272</td>
<td>4.575</td>
</tr>
<tr>
<td>Clairemont Mesa</td>
<td>3.737297</td>
<td>0.427366</td>
<td>8.745</td>
</tr>
<tr>
<td>College Area</td>
<td>3.370263</td>
<td>0.413479</td>
<td>8.151</td>
</tr>
<tr>
<td>Core</td>
<td>3.635188</td>
<td>0.490640</td>
<td>7.409</td>
</tr>
<tr>
<td>Cortez Hill</td>
<td>1.631032</td>
<td>0.299654</td>
<td>5.443</td>
</tr>
<tr>
<td>Del Mar Heights</td>
<td>1.346206</td>
<td>0.342628</td>
<td>3.927</td>
</tr>
<tr>
<td>East Village</td>
<td>2.600489</td>
<td>0.190932</td>
<td>13.620</td>
</tr>
<tr>
<td>Gaslamp Quarter</td>
<td>3.183092</td>
<td>0.527615</td>
<td>6.033</td>
</tr>
<tr>
<td>Grant Hill</td>
<td>2.770976</td>
<td>0.416838</td>
<td>6.648</td>
</tr>
<tr>
<td>Grantville</td>
<td>2.177175</td>
<td>0.693599</td>
<td>3.139</td>
</tr>
<tr>
<td>Kensington</td>
<td>1.284044</td>
<td>0.671482</td>
<td>1.912</td>
</tr>
<tr>
<td>La Jolla</td>
<td>0.852667</td>
<td>0.099413</td>
<td>8.577</td>
</tr>
<tr>
<td>La Jolla Village</td>
<td>0.984426</td>
<td>1.193870</td>
<td>0.825</td>
</tr>
<tr>
<td>Linda Vista</td>
<td>2.359895</td>
<td>0.393392</td>
<td>5.999</td>
</tr>
<tr>
<td>Little Italy</td>
<td>2.600567</td>
<td>0.275834</td>
<td>9.428</td>
</tr>
<tr>
<td>Loma Portal</td>
<td>2.575164</td>
<td>0.249679</td>
<td>10.314</td>
</tr>
</tbody>
</table>
### 6.5. SPATIAL HETEROGENEITY

<table>
<thead>
<tr>
<th>Neighborhood</th>
<th>Bathrooms</th>
<th>Neighborhood</th>
<th>Latitude</th>
<th>Longitude</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marina</td>
<td>3.317139</td>
<td>0.656533</td>
<td>5.053</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Midtown</td>
<td>0.899736</td>
<td>0.112205</td>
<td>8.019</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Midtown District</td>
<td>3.143440</td>
<td>0.594875</td>
<td>5.284</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mira Mesa</td>
<td>2.858280</td>
<td>0.512511</td>
<td>5.577</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mission Bay</td>
<td>1.764929</td>
<td>0.122421</td>
<td>14.417</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mission Valley</td>
<td>2.666000</td>
<td>1.365483</td>
<td>1.952</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moreno Mission</td>
<td>3.234512</td>
<td>0.557988</td>
<td>5.798</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal Heights</td>
<td>3.505139</td>
<td>0.467965</td>
<td>7.490</td>
<td></td>
<td></td>
</tr>
<tr>
<td>North Clairemont</td>
<td>2.574847</td>
<td>0.613471</td>
<td>4.197</td>
<td></td>
<td></td>
</tr>
<tr>
<td>North Hills</td>
<td>2.584724</td>
<td>0.191541</td>
<td>13.494</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ocean Beach</td>
<td>1.702208</td>
<td>0.207508</td>
<td>8.203</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Old Town</td>
<td>2.249120</td>
<td>0.302755</td>
<td>7.429</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Otay Ranch</td>
<td>2.818736</td>
<td>1.132794</td>
<td>2.488</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pacific Beach</td>
<td>2.272803</td>
<td>0.130607</td>
<td>17.402</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Park West</td>
<td>2.676739</td>
<td>0.308257</td>
<td>8.683</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rancho Bernadino</td>
<td>0.856723</td>
<td>0.555198</td>
<td>1.543</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rancho Penasquitos</td>
<td>0.677767</td>
<td>0.414569</td>
<td>1.635</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roseville</td>
<td>1.109625</td>
<td>0.360103</td>
<td>3.081</td>
<td></td>
<td></td>
</tr>
<tr>
<td>San Carlos</td>
<td>2.489815</td>
<td>0.511232</td>
<td>4.870</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scripps Ranch</td>
<td>2.459862</td>
<td>0.469601</td>
<td>5.238</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serra Mesa</td>
<td>2.968934</td>
<td>0.602807</td>
<td>4.925</td>
<td></td>
<td></td>
</tr>
<tr>
<td>South Park</td>
<td>2.895471</td>
<td>0.521793</td>
<td>5.549</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University City</td>
<td>3.125387</td>
<td>0.347825</td>
<td>8.986</td>
<td></td>
<td></td>
</tr>
<tr>
<td>West University Heights</td>
<td>2.188257</td>
<td>0.390408</td>
<td>5.605</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Balboa Park</td>
<td>0.605655</td>
<td>0.245384</td>
<td>2.468</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bay Ho</td>
<td>0.836163</td>
<td>0.631871</td>
<td>1.323</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bay Park</td>
<td>1.060944</td>
<td>0.430737</td>
<td>2.463</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carmel Valley</td>
<td>0.521954</td>
<td>0.480497</td>
<td>1.086</td>
<td></td>
<td></td>
</tr>
<tr>
<td>City Heights West</td>
<td>-0.272600</td>
<td>0.663983</td>
<td>-0.411</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clairemont Mesa</td>
<td>-0.742539</td>
<td>0.450344</td>
<td>-1.649</td>
<td></td>
<td></td>
</tr>
<tr>
<td>College Area</td>
<td>-0.306621</td>
<td>0.410476</td>
<td>-0.747</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>-0.786470</td>
<td>0.395991</td>
<td>-1.986</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cortez Hill</td>
<td>0.793039</td>
<td>0.380195</td>
<td>2.086</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Del Mar Heights</td>
<td>-0.071069</td>
<td>0.369070</td>
<td>-0.193</td>
<td></td>
<td></td>
</tr>
<tr>
<td>East Village</td>
<td>-0.186076</td>
<td>0.213572</td>
<td>-0.871</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaslamp Quarter</td>
<td>-0.294024</td>
<td>0.342057</td>
<td>-0.860</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grant Hill</td>
<td>-0.456825</td>
<td>0.425374</td>
<td>-1.074</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grantville</td>
<td>0.907259</td>
<td>0.770945</td>
<td>1.177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kensington</td>
<td>-0.257195</td>
<td>1.009326</td>
<td>-0.255</td>
<td></td>
<td></td>
</tr>
<tr>
<td>La Jolla</td>
<td>-0.152098</td>
<td>0.133726</td>
<td>-1.137</td>
<td></td>
<td></td>
</tr>
<tr>
<td>La Jolla Village</td>
<td>4.291700</td>
<td>1.882046</td>
<td>2.280</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linda Vista</td>
<td>-0.485372</td>
<td>0.642684</td>
<td>-0.755</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Little Italy</td>
<td>0.057475</td>
<td>0.306357</td>
<td>8.188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loma Portal</td>
<td>-0.406484</td>
<td>0.250607</td>
<td>-1.622</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marina Portal</td>
<td>-0.831114</td>
<td>0.511626</td>
<td>-1.624</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## bedrooms:neighborhoodMidtown 0.696852 0.167900 4.150
## bedrooms:neighborhoodMidtown District 0.010614 0.509151 0.021
## bedrooms:neighborhoodMira Mesa -0.197692 0.780959 -0.253
## bedrooms:neighborhoodMission Bay -0.330540 0.121602 -2.718
## bedrooms:neighborhoodMission Valley 0.514998 1.295767 0.397
## bedrooms:neighborhoodMoreno Mission -0.584689 0.596044 -0.981
## bedrooms:neighborhoodNormal Heights -0.127744 0.391691 -0.326
## bedrooms:neighborhoodNorth Clairemont 0.281306 0.695297 0.405
## bedrooms:neighborhoodNorth Hills 0.380444 0.178477 2.132
## bedrooms:neighborhoodNorthwest 0.288603 0.607295 0.475
## bedrooms:neighborhoodOcean Beach -0.038069 0.139026 -0.273
## bedrooms:neighborhoodOld Town -0.319724 0.375203 -0.852
## bedrooms:neighborhoodOtay Ranch 0.015564 1.332279 0.012
## bedrooms:neighborhoodPacific Beach -0.037912 0.139026 -0.273
## bedrooms:neighborhoodPark West -0.696514 0.413881 -1.683
## bedrooms:neighborhoodRancho Bernadino 1.034776 0.579798 1.875
## bedrooms:neighborhoodRancho Penasquitos 0.674520 0.519260 1.299
## bedrooms:neighborhoodRoseville 0.881011 0.592662 1.486
## bedrooms:neighborhoodSan Carlos -0.394191 0.540343 -0.730
## bedrooms:neighborhoodScripps Ranch 1.107455 0.336101 3.295
## bedrooms:neighborhoodSerra Mesa 0.253001 0.620774 0.408
## bedrooms:neighborhoodSouth Park -0.595844 0.407811 -1.461
## bedrooms:neighborhoodUniversity City 0.203783 0.455767 0.447
## bedrooms:neighborhoodWest University Heights 0.242873 0.359245 0.676
## beds:neighborhoodBalboa Park 0.041556 0.173183 0.240
## beds:neighborhoodBay Ho -0.402544 0.495241 -0.813
## beds:neighborhoodBay Park 0.283958 0.410776 0.691
## beds:neighborhoodCarmel Valley 0.150416 0.288268 0.522
## beds:neighborhoodCore 0.602559 0.277027 2.175
## beds:neighborhoodCortez Hill -0.609996 0.143559 -4.249
## beds:neighborhoodDel Mar Heights -0.708476 0.257299 -2.754
## beds:neighborhoodEast Village 0.399909 0.148641 2.690
## beds:neighborhoodGaslamp Quarter 0.240245 0.319910 0.751
## beds:neighborhoodGrant Hill -1.315807 0.186724 -7.047
## beds:neighborhoodGrantville -0.382590 0.469011 -0.816
## beds:neighborhoodKensington 0.134374 0.664968 0.201
## beds:neighborhoodLa Jolla 0.001347 0.085013 0.016
## beds:neighborhoodLa Jolla Village -2.878676 1.020652 -2.820
## beds:neighborhoodLinda Vista -0.142372 0.278211 -0.512
## beds:neighborhoodLittle Italy -0.569868 0.099961 -5.701
## beds:neighborhoodLoma Portal -0.255510 0.222956 -1.146
## beds:neighborhoodMarina 0.024175 0.429466 0.056
## beds:neighborhoodMidtown -0.346866 0.137915 -2.515
## beds:neighborhoodMidtown District -0.464781 0.337775 -1.376
## beds:neighborhoodMira Mesa 0.319934 0.426799 0.750
## beds:neighborhoodMission Bay -0.108936 0.067105 -1.623
## beds:neighborhoodMission Valley -0.502441 0.879795 -0.571
## beds:neighborhoodMoreno Mission 0.492514 0.439355 1.121
## beds:neighborhoodNormal Heights -0.532907 0.227211 -2.345
## beds:neighborhoodNorth Clairemont 0.562363 0.704213 0.799
## beds:neighborhoodNorth Hills -0.279430 0.123678 -2.259
## beds:neighborhoodNorthwest 0.742017 0.474903 1.562
## beds:neighborhoodOcean Beach -0.667651 0.137647 -4.850
## beds:neighborhoodOld Town 0.459210 0.287008 1.600
## beds:neighborhoodOtay Ranch 0.235723 0.983870 0.240
## beds:neighborhoodPacific Beach -0.179242 0.087511 -2.048
## beds:neighborhoodPark West -0.873297 0.225334 -3.876
## beds:neighborhoodRancho Bernadino 0.378088 0.348640 1.084
## beds:neighborhoodRancho Penasquitos 0.147457 0.344820 0.428
## beds:neighborhoodRoseville -0.391529 0.328609 -1.191
## beds:neighborhoodSan Carlos 0.115338 0.621666 0.186
## beds:neighborhoodScripps Ranch -1.654484 0.338331 -4.890
## beds:neighborhoodSerra Mesa -1.018812 0.705888 -1.443
## beds:neighborhoodSouth Park 0.452815 0.406052 1.115
## beds:neighborhoodUniversity City -0.345822 0.232779 -1.486
## beds:neighborhoodWest University Heights 0.146128 0.364075 0.401
## Pr(>|t|)
## accommodates:neighborhoodBalboa Park 0.495668
## accommodates:neighborhoodBay Ho 0.438397
## accommodates:neighborhoodBay Park 0.126774
## accommodates:neighborhoodCarmel Valley 0.488131
## accommodates:neighborhoodCity Heights West 0.053861
## accommodates:neighborhoodClaremont Mesa 6.40e-05 ***
## accommodates:neighborhoodCollege Area 0.065740
## accommodates:neighborhoodCore 0.396446
## accommodates:neighborhoodCortez Hill 1.61e-08 ***
## accommodates:neighborhoodDel Mar Heights 0.000108 ***
## accommodates:neighborhoodEast Village 0.005665 **
## accommodates:neighborhoodGaslamp Quarter 0.022395 *
## accommodates:neighborhoodGrant Hill 1.45e-11 ***
## accommodates:neighborhoodGrantville 0.283202
## accommodates:neighborhoodKensington 0.137520
## accommodates:neighborhoodLa Jolla < 2e-16 ***
## accommodates:neighborhoodLa Jolla Village 0.170446
## accommodates:neighborhoodLinda Vista 0.063414
## accommodates:neighborhoodLittle Italy 7.47e-07 ***
## accommodates:neighborhoodLoma Portal 0.000138 ***
## accommodates:neighborhoodMarina 0.012484 *
## accommodates:neighborhoodMidtown 2.40e-12 ***
<table>
<thead>
<tr>
<th>Neighborhood</th>
<th>Accommodates Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midtown District</td>
<td>0.024780 *</td>
</tr>
<tr>
<td>Mira Mesa</td>
<td>0.953213</td>
</tr>
<tr>
<td>Mission Bay</td>
<td>&lt; 2e-16 ***</td>
</tr>
<tr>
<td>Mission Valley</td>
<td>0.776000</td>
</tr>
<tr>
<td>Moreno Mission</td>
<td>0.661505</td>
</tr>
<tr>
<td>Normal Heights</td>
<td>0.037279 *</td>
</tr>
<tr>
<td>North Clairemont</td>
<td>0.429327</td>
</tr>
<tr>
<td>North Hills</td>
<td>0.001602 **</td>
</tr>
<tr>
<td>Northwest</td>
<td>0.370103</td>
</tr>
<tr>
<td>Ocean Beach</td>
<td>&lt; 2e-16 ***</td>
</tr>
<tr>
<td>Old Town</td>
<td>0.267333</td>
</tr>
<tr>
<td>Otay Ranch</td>
<td>0.281298</td>
</tr>
<tr>
<td>Pacific Beach</td>
<td>2.12e-09 ***</td>
</tr>
<tr>
<td>Park West</td>
<td>5.96e-09 ***</td>
</tr>
<tr>
<td>Rancho Bernadino</td>
<td>0.643202</td>
</tr>
<tr>
<td>Rancho Penasquitos</td>
<td>0.593817</td>
</tr>
<tr>
<td>Roseville</td>
<td>0.161071</td>
</tr>
<tr>
<td>San Carlos</td>
<td>0.548479</td>
</tr>
<tr>
<td>Scripps Ranch</td>
<td>0.006311 **</td>
</tr>
<tr>
<td>Serra Mesa</td>
<td>0.079258</td>
</tr>
<tr>
<td>South Park</td>
<td>0.192775</td>
</tr>
<tr>
<td>University City</td>
<td>0.344762</td>
</tr>
<tr>
<td>West University Heights</td>
<td>0.369719</td>
</tr>
<tr>
<td>Balboa Park</td>
<td>&lt; 2e-16 ***</td>
</tr>
<tr>
<td>Bay Ho</td>
<td>4.60e-10 ***</td>
</tr>
<tr>
<td>Bay Park</td>
<td>1.11e-09 ***</td>
</tr>
<tr>
<td>Carmel Valley</td>
<td>1.11e-07 ***</td>
</tr>
<tr>
<td>City Heights West</td>
<td>4.87e-06 ***</td>
</tr>
<tr>
<td>Clairemont Mesa</td>
<td>&lt; 2e-16 ***</td>
</tr>
<tr>
<td>College Area</td>
<td>4.37e-16 ***</td>
</tr>
<tr>
<td>Core</td>
<td>1.45e-13 ***</td>
</tr>
<tr>
<td>Cortez Hill</td>
<td>5.45e-08 ***</td>
</tr>
<tr>
<td>Del Mar Heights</td>
<td>8.71e-05 ***</td>
</tr>
<tr>
<td>East Village</td>
<td>&lt; 2e-16 ***</td>
</tr>
<tr>
<td>Gaslamp Quarter</td>
<td>1.71e-09 ***</td>
</tr>
<tr>
<td>Grant Hill</td>
<td>3.25e-11 ***</td>
</tr>
<tr>
<td>Grantville</td>
<td>0.001704 **</td>
</tr>
<tr>
<td>Kensington</td>
<td>0.055892</td>
</tr>
<tr>
<td>La Jolla</td>
<td>&lt; 2e-16 ***</td>
</tr>
<tr>
<td>La Jolla Village</td>
<td>0.409651</td>
</tr>
<tr>
<td>Linda Vista</td>
<td>2.10e-09 ***</td>
</tr>
<tr>
<td>Little Italy</td>
<td>&lt; 2e-16 ***</td>
</tr>
<tr>
<td>Loma Portal</td>
<td>&lt; 2e-16 ***</td>
</tr>
<tr>
<td>Marina</td>
<td>4.49e-07 ***</td>
</tr>
<tr>
<td>Midtown</td>
<td>1.28e-15 ***</td>
</tr>
<tr>
<td>Midtown District</td>
<td>1.31e-07 ***</td>
</tr>
</tbody>
</table>
## bathrooms:neighborhoodMira Mesa 2.55e-08 ***
## bathrooms:neighborhoodMission Bay < 2e-16 ***
## bathrooms:neighborhoodMission Valley 0.050935 .
## bathrooms:neighborhoodMoreno Mission 7.07e-09 ***
## bathrooms:neighborhoodNormal Heights 7.88e-14 ***
## bathrooms:neighborhoodNorth Clairemont 2.74e-05 ***
## bathrooms:neighborhoodNorth Hills < 2e-16 ***
## bathrooms:neighborhoodNorthwest 4.58e-07 ***
## bathrooms:neighborhoodOcean Beach 2.85e-16 ***
## bathrooms:neighborhoodOld Town 1.25e-13 ***
## bathrooms:neighborhoodOtay Ranch 0.012863 *
## bathrooms:neighborhoodPacific Beach < 2e-16 ***
## bathrooms:neighborhoodPark West < 2e-16 ***
## bathrooms:neighborhoodRancho Bernadino 0.122861
## bathrooms:neighborhoodRancho Penasquitos 0.102129
## bathrooms:neighborhoodRoseville 0.002070 **
## bathrooms:neighborhoodSan Carlos 1.14e-06 ***
## bathrooms:neighborhoodScripps Ranch 1.68e-07 ***
## bathrooms:neighborhoodSerra Mesa 8.66e-07 ***
## bathrooms:neighborhoodSouth Park 3.00e-08 ***
## bathrooms:neighborhoodUniversity City < 2e-16 ***
## bathrooms:neighborhoodWest University Heights 2.18e-08 ***
## bedrooms:neighborhoodBalboa Park 0.013608 *
## bedrooms:neighborhoodBay Ho 0.185782
## bedrooms:neighborhoodBay Park 0.013803 *
## bedrooms:neighborhoodCarmel Valley 0.277400
## bedrooms:neighborhoodCity Heights West 0.681416
## bedrooms:neighborhoodClairemont Mesa 0.099236 .
## bedrooms:neighborhoodCollege Area 0.455100
## bedrooms:neighborhoodCore 0.047070 *
## bedrooms:neighborhoodCoretza Hill 0.037033 *
## bedrooms:neighborhoodDel Mar Heights 0.847309
## bedrooms:neighborhoodEast Village 0.383650
## bedrooms:neighborhoodGaslamp Quarter 0.390058
## bedrooms:neighborhoodGrant Hill 0.282895
## bedrooms:neighborhoodGrantville 0.239317
## bedrooms:neighborhoodKensington 0.798872
## bedrooms:neighborhoodLa Jolla 0.255422
## bedrooms:neighborhoodLa Jolla Village 0.022623 *
## bedrooms:neighborhoodLinda Vista 0.450143
## bedrooms:neighborhoodLittle Italy 0.851191
## bedrooms:neighborhoodLoma Portal 0.104857
## bedrooms:neighborhoodMarina 0.104332
## bedrooms:neighborhoodMidtown 3.37e-05 ***
## bedrooms:neighborhoodMidtown District 0.983369
## bedrooms:neighborhoodMira Mesa 0.800169
## bedrooms:neighborhoodMission Bay 0.006583 **
## bedrooms:neighborhoodMission Valley 0.691053
## bedrooms:neighborhood Moreno Mission 0.326658
## bedrooms:neighborhood Normal Heights 0.744334
## bedrooms:neighborhood North Clairemont 0.685798
## bedrooms:neighborhood North Hills 0.033080 *
## bedrooms:neighborhood Northwest 0.634643
## bedrooms:neighborhood Ocean Beach 0.854736
## bedrooms:neighborhood Old Town 0.394173
## bedrooms:neighborhood Otay Ranch 0.990680
## bedrooms:neighborhood Pacific Beach 0.785097
## bedrooms:neighborhood Park West 0.092450 .
## bedrooms:neighborhood Rancho Bernadino 0.074358 .
## bedrooms:neighborhood Rancho Penasquitos 0.193994
## bedrooms:neighborhood Roseville 0.137390
## bedrooms:neighborhood San Carlos 0.465713
## bedrooms:neighborhood Scripps Ranch 0.000990 ***
## bedrooms:neighborhood Serra Mesa 0.683614
## bedrooms:neighborhood South Park 0.144046
## bedrooms:neighborhood University City 0.654804
## bedrooms:neighborhood West University Heights 0.499025
## beds:neighborhood Balboa Park 0.810374
## beds:neighborhood Bay Ho 0.416352
## beds:neighborhood Bay Park 0.489423
## beds:neighborhood Carmel Valley 0.601836
## beds:neighborhood City Heights West 0.682195
## beds:neighborhood Clairemont Mesa 0.000332 ***
## beds:neighborhood College Area 0.057226 .
## beds:neighborhood Core 0.029663 *
## beds:neighborhood Cortez Hill 2.18e-05 ***
## beds:neighborhood Del Mar Heights 0.005914 **
## beds:neighborhood East Village 0.007156 **
## beds:neighborhood Gaslamp Quarter 0.452696
## beds:neighborhood Grant Hill 2.04e-12 ***
## beds:neighborhood Grantville 0.414683
## beds:neighborhood Kensington 0.840859
## beds:neighborhood La Jolla 0.987357
## beds:neighborhood La Jolla Village 0.004812 **
## beds:neighborhood Linda Vista 0.608851
## beds:neighborhood Little Italy 1.25e-08 ***
## beds:neighborhood Loma Portal 0.251837
## beds:neighborhood Marina 0.955112
## beds:neighborhood Midtown 0.011927 *
## beds:neighborhood Midtown District 0.168872
## beds:neighborhood Mira Mesa 0.453518
## beds:neighborhood Mission Bay 0.104565
### 6.6. Spatial Dependence

As we have just discussed, SH is about effects of phenomena that are explicitly linked to geography and that hence cause spatial variation and clustering of values. This encompasses many of the kinds of spatial effects we may be interested in when we fit linear regressions. However, in other cases, our interest is on the effect of the spatial configuration of the observations, and the extent to which that has an effect on the outcome we are considering. For example, we might think that the price of a house not only depends on the number of bathrooms it has but, if we take number of bathrooms as a proxy for size and status, also whether it is surrounded by other houses with many bathrooms. This kind of spatial effect is fundamentally different from SH in that it is not related to inherent characteristics of the geography but relates to the characteristics of the observations in our dataset and, specially, to their spatial arrangement. We call this phenomenon by which the values of observations are related to each other for every neighborhood.
through distance *spatial dependence* (Anselin, 1988).

**Spatial Weights**

There are several ways to introduce spatial dependence in an econometric framework, with varying degrees of econometric sophistication (see Anselin, 2003, for a good overview). Common to all of them however is the way space is formally encapsulated: through *spatial weights matrices* ($W$)\(^3\). These are $N \times N$ matrices with zero diagonals and every $w_{ij}$ cell with a value that represents the degree of spatial connectivity/interaction between observations $i$ and $j$. If they are not connected at all, $w_{ij} = 0$, otherwise $w_{ij} > 0$ and we call $i$ and $j$ neighbors. The exact value in the latter case depends on the criterium we use to define neighborhood relations. These matrices also tend to be row-standardized so the sum of each row equals to one.

A related concept to spatial weight matrices is that of *spatial lag*. This is an operator that multiplies a given variable $y$ by a spatial weight matrix:

$$y_{lag} = Wy$$

If $W$ is row-standardized, $y_{lag}$ is effectively the average value of $y$ in the neighborhood of each observation. The individual notation may help clarify this:

$$y_{lag-i} = \sum_j w_{ij}y_j$$

where $y_{lag-i}$ is the spatial lag of variable $y$ at location $i$, and $j$ sums over the entire dataset. If $W$ is row-standardized, $y_{lag-i}$ becomes an average of $y$ weighted by the spatial criterium defined in $W$.

Given that spatial weights matrices are not the focus of this tutorial, we will stick to a very simple case. Since we are dealing with points, we will use $K$-nn weights, which take the $k$ nearest neighbors of each observation as neighbors and assign a value of one, assigning everyone else a zero. We will use $k = 50$ to get a good degree of variation and sensible results.

```r
# Create knn list of each house
hnn <- db %>%
  st_coordinates() %>%
  as.matrix() %>%
  knearneigh(k = 50)
# Create nb object
hnb <- knn2nb(hnn)
# Create spatial weights matrix (note it row-standardizes by default)
hknn <- nb2listw(hnb)
```

\(^3\)If you need to refresh your knowledge on spatial weight matrices, check Block E of Arribas-Bel (2019) or Chapter 4 of Rey et al. (ming).
We can inspect the weights created by simply typing the name of the object:

```
hknn
```

```r
## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 6110
## Number of nonzero links: 305500
## Percentage nonzero weights: 0.8183306
## Average number of links: 50
## Non-symmetric neighbours list
##
## Weights style: W
## Weights constants summary:
## n   nn  S0   S1   S2
## W 6110 37332100 6110 220.5032 24924.44
```

### Exogenous spatial effects

Let us come back to the house price example we have been working with. So far, we have hypothesized that the price of an AirBnb property in San Diego can be explained using information about its own characteristics, and the neighbourhood it belongs to. However, we can hypothesise that the price of a house is also affected by the characteristics of the houses surrounding it. Considering it as a proxy for larger and more luxurious houses, we will use the number of bathrooms of neighboring houses as an additional explanatory variable. This represents the most straightforward way to introduce spatial dependence in a regression, by considering not only a given explanatory variable, but also its spatial lag.

In our example case, in addition to including the number of bathrooms of the property, we will include its spatial lag. In other words, we will be saying that it is not only the number of bathrooms in a house but also that of the surrounding properties that helps explain the final price at which a house is advertised for. Mathematically, this implies estimating the following model:

\[
\log(P_i) = \alpha + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + \beta_5 Bath_{lag-i} + \epsilon_i
\]

Let us first compute the spatial lag of `imd_score`:

```
db$w_bathrooms <- lag.listw(hknn, db$bathrooms)
```

And then we can include it in our previous specification. Note that we apply the log to the lag, not the reverse:

```
m5 <- lm(log_price ~ accommodates + bedrooms + beds + bathrooms + w_bathrooms, db)
```
As we can see, the lag is not only significative and positive, but its effect seems to be even larger than that of the property itself. Taken literally, this implies that the average number of bathrooms in AirBnb's nearby has a larger effect on the final price of a given AirBnb than its own number of bathrooms. There are several ways to interpret this. One is that, if we take the spatial lag of bathrooms, as we said above, to be a proxy for the types of houses surrounding a property, this is probably a better predictor of how wealthy an area is than the number of bathrooms of a single property, which is more variable. If we also assume that the area where an AirBnb is located has a bigger effect on price than the number of bathrooms, we can start seeing an answer to the apparent puzzle.

A note on more advanced spatial regression

Introducing a spatial lag of an explanatory variable, as we have just seen, is the most straightforward way of incorporating the notion of spatial dependence in a linear regression framework. It does not require additional changes, it can be estimated with OLS, and the interpretation is rather similar to interpreting non-spatial variables. The field of spatial econometrics however is a much broader
6.7. PREDICTING HOUSE PRICES

one and has produced over the last decades many techniques to deal with spatial effects and spatial dependence in different ways. Although this might be an over simplification, one can say that most of such efforts for the case of a single cross-section are focused on two main variations: the spatial lag and the spatial error model. Both are similar to the case we have seen in that they are based on the introduction of a spatial lag, but they differ in the component of the model they modify and affect.

The spatial lag model introduces a spatial lag of the dependent variable. In the example we have covered, this would translate into:

\[
\log(P_i) = \alpha + \rho \log(P_i) + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + \epsilon_i
\]

Although it might not seem very different from the previous equation, this model violates the exogeneity assumption, crucial for OLS to work.

Equally, the spatial error model includes a spatial lag in the error term of the equation:

\[
\log(P_i) = \alpha + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + u_i
\]

\[
u_i = u_{lag-i} + \epsilon_i
\]

Again, although similar, one can show this specification violates the assumptions about the error term in a classical OLS model.

Both the spatial lag and error model violate some of the assumptions on which OLS relies and thus render the technique unusable. Much of the efforts have thus focused on coming up with alternative methodologies that allow unbiased, robust, and efficient estimation of such models. A survey of those is beyond the scope of this note, but the interested reader is referred to Anselin (1988), Anselin (2003), and Anselin and Rey (2014) for further reference.

6.7 Predicting house prices

So far, we have seen how to exploit the output of a regression model to evaluate the role different variables play in explaining another one of interest. However, once fit, a model can also be used to obtain predictions of the dependent variable given a new set of values for the explanatory variables. We will finish this session by dipping our toes in predicting with linear models.

The core idea is that once you have estimates for the way in which the explanatory variables can be combined to explain the dependent one, you can plug new values on the explanatory side of the model and combine them following
the model estimates to obtain predictions. In the example we have worked with, you can imagine this application would be useful to obtain valuations of a house, given we know its characteristics.

Conceptually, predicting in linear regression models involves using the estimates of the parameters to obtain a value for the dependent variable:

$$\log(P_i) = \bar{\alpha} + \bar{\beta}_1 Acc_i^* + \bar{\beta}_2 Bath_i^* + \bar{\beta}_3 Bedr_i^* + \bar{\beta}_4 Beds_i^*$$

where $\log P_i$ is our predicted value, and we include the $\bar{}$ sign to note that it is our estimate obtained from fitting the model. We use the $^*$ sign to note that those can be new values for the explanatory variables, not necessarily those used to fit the model.

Technically speaking, prediction in linear models is relatively streamlined in R. Suppose we are given data for a new house which is to be put on the AirBnb platform. We know it accommodates four people, and has two bedrooms, three beds, and one bathroom. We also know that the surrounding properties have, on average, 1.5 bathrooms. Let us record the data first:

```r
new.house <- data.frame(
    accommodates = 4,
    bedrooms = 2,
    beds = 3,
    bathrooms = 1,
    w_bathrooms = 1.5
)
```

To obtain the prediction for its price, we can use the `predict` method:

```r
new.price <- predict(m5, new.house)
new.price
```

```
## 1
## 4.900168
```

Now remember we were using the log of the price as dependent variable. If we want to recover the actual price of the house, we need to take its exponent:

```r
exp(new.price)
```

```
## 1
## 134.3123
```

According to our model, the house would be worth $134.3123448.

### 6.8 Questions

We will be using again the Madrid AirBNb dataset:
In addition to those we have already seen, the columns to use here are:

- **neighbourhood**: a column with the name of the neighbourhood in which the property is located

With this at hand, answer the following questions:

1. Fit a baseline model with only property characteristics explaining the log of price

\[
\log(P_i) = \alpha + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + \epsilon_i
\]

2. Augment the model with fixed effects at the neighbourhood level

\[
\log(P_i) = \alpha_r + \beta_1 Acc_i + \beta_2 Bath_i + \beta_3 Bedr_i + \beta_4 Beds_i + \epsilon_i
\]

3. [Optional] Augment the model with spatial regimes at the neighbourhood level:

\[
\log(P_i) = \alpha_r + \beta_{r1} Acc_i + \beta_{r2} Bath_i + \beta_{r3} Bedr_i + \beta_{r4} Beds_i + \epsilon_{r,i}
\]

4. Fit a model that augments the baseline in 1. with the spatial lag of a variable you consider interesting. Motivate this choice. Note that to complete this, you will need to also generate a spatial weights matrix.

In each instance, provide a brief interpretation (no more than a few lines for each) that demonstrates your understanding of the underlying concepts behind your approach.
Chapter 7

Multilevel Modelling - Part 1

This chapter provides an introduction to multi-level data structures and multi-level modelling and draws on the following references:

- Gelman and Hill (2006b) provides an excellent and intuitive explanation of multilevel modelling and data analysis in general. Read Part 2A for a really good explanation of multilevel models.
- for Multilevel Modelling (nd) is an useful online resource on multilevel modelling and is free!

7.1 Dependencies

This chapter uses the following libraries which are listed in the Dependency list Section of Chapter 1:

```r
# Data manipulation, transformation and visualisation
library(tidyverse)

# Nice tables
library(kableExtra)

# Simple features (a standardised way to encode vector data ie. points, lines, polygons)
library(sf)

# Spatial objects conversion
library(sp)

# Thematic maps
library(tmap)
```

119
# Colour palettes
library(RColorBrewer)
# More colour palettes
library(viridis)
# Nice colour schemes
# Fitting multilevel models
library(lme4)
# Tools for extracting information generated by lme4
library(merTools)
# Exportable regression tables
library(jtools)
library(stargazer)
library(sjPlot)

## 7.2 Data

For this chapter, we will use data for Liverpool from England’s 2011 Census. The original source is the Office of National Statistics and the dataset comprises a number of selected variables capturing demographic, health and socio-economic attributes of the local resident population at four geographic levels: Output Area (OA), Lower Super Output Area (LSOA), Middle Super Output Area (MSOA) and Local Authority District (LAD). The variables include population counts and percentages. For a description of the variables, see the readme file in the mlm data folder.¹

Let us read the data:

```r
# clean workspace
rm(list=ls())
# read data
oa_shp <- st_read("data/mlm/OA.shp")
```

We can now attach and visualise the structure of the data.

```r
# attach data frame
attach(oa_shp)

# sort data by oa
oa_shp <- oa_shp[order(oa_cd),]
head(oa_shp)
```

```r
## Simple feature collection with 6 features and 19 fields
## geometry type: MULTIPOLYGON
## dimension:     XY
## bbox:           xmin: 335056  ymin: 389163  xmax: 336155  ymax: 389642
## projected CRS: Transverse_Mercator
```

¹Read the file in R by executing `read_tsv("data/mlm/readme.txt")`
### Data Structure

The data are hierarchically structured: OAs nested within LSOAs; LSOAs nested within MSOAs; and, MSOAs nested within LADs. Observations nested within higher geographical units may be correlated.

This is one type of hierarchical structure. There is a range of data structures:

- Strict nested data structures eg. an individual unit is nested within only one higher unit
- Repeated measures structures eg. various measurements for an individual

#### Figure 7.1: Fig. 1. Data Structure.
• Crossed classified structures eg. individuals may work and live in different neighbourhoods

• Multiple membership structure eg. individuals may have two different work places

**Why should we care about the structure of the data?**

• *Draw correct statistical inference*: Failing to recognise hierarchical structures will lead to underestimated standard errors of regression coefficients and an overstatement of statistical significance. Standard errors for the coefficients of higher-level predictor variables will be the most affected by ignoring grouping.

• *Link context to individual units*: We can link and understand the extent of group effects on individual outcomes eg. how belonging to a certain socio-economic group influences on future career opportunities.

• *Spatial dependency*: Recognising the hierarchical structure of data may help mitigate the effects of severe spatial autocorrelation.

Quickly, let us get a better idea about the data and look at the number of OAs nested within LSOAs and MSOAs

```r
# mean of nested OAs within LSOAs and MSOAs
lsoa_cd %>% table() %>% mean() %>% round(, 2)
## [1] 5

msoa_cd %>% table() %>% mean() %>% round(, 2)
## [1] 26

# number of OAs nested within LSOAs and MSOAs
lsoa_cd %>% table() %>% sort() %>% plot()
```
7.2. DATA

```r
msoa_cd %>% table() %>%
  sort() %>%
  plot()
```
7.3 Modelling

We should now be persuaded that ignoring the hierarchical structure of data may be a major issue. Let us now use a simple example to understand the intuition of multilevel model using the census data. We will seek to understand the spatial distribution of the proportion of population in unemployment in Liverpool, particularly why and where concentrations in this proportion occur. To illustrate the advantages of taking a multilevel modelling approach, we will start by estimating a linear regression model and progressively building complexity. We will first estimate a model and then explain the intuition underpinning the process. We will seek to gain a general understanding of multilevel modelling. If you are interested in the statistical and mathematical formalisation of the underpinning concepts, please refer to Gelman and Hill (2006b).

We first need to want to understand our dependent variable: its density distribution:

```r
ggplot(data = oa_shp) + geom_density(alpha=0.8, colour="black", fill="lightblue", aes(x = unemp)) + theme_classic()
```

![Graph showing density distribution of unemployment.](image)

```r
summary(unemp)
```

```r
#>  Min. 1st Qu.  Median    Mean 3rd Qu.   Max.   
#> 0.00000 0.05797 0.10256 0.11581 0.16129 0.50000
```

and, its spatial distribution:
Let us look at those areas:

```r
# high %
oa_shp %>% filter(unemp > 0.2) %>%
  dplyr::select(oa_cd, pop, unemp)
```

```
## Simple feature collection with 203 features and 3 fields
## geometry type:  GEOMETRY
## dimension:      XY
## bbox:           xmin: 333993.8 ymin: 379748.5 xmax: 345600.2 ymax: 397681.5
## projected CRS: Transverse_Mercator
## First 10 features:
## 1  E00032992 240 0.2519685 MULTIPOLYGON (((335804.9 3...
## 7.3.1 Baseline Linear Regression Model

Now let us estimate a simple linear regression model with the intercept only:

```r
# specify a model equation
eq1 <- unemp ~ 1
model1 <- lm(formula = eq1, data = oa_shp)

# estimates
summary(model1)
```

```
## Call:
## lm(formula = eq1, data = oa_shp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##-0.11581 -0.05784 -0.01325  0.04548  0.38419
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.115812   0.001836   63.09  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07306 on 1583 degrees of freedom
```

To understand the differences between the linear regression model and multilevel models, let us consider the model we have estimated:

\[ y_i = \beta_0 + e_i \]

where \( y_i \) represents the proportion of the unemployed resident population in the OA \( i \); \( \beta_0 \) is the regression intercept and measures the average proportion of the unemployed resident population across OAs; and, \( e_i \) is the error term. But how do we deal with the hierarchical structure of the data?
7.3.1.1 Limitations

Before looking at the answer, let’s first understand some of the key limitations of the linear regression model to handle the hierarchical structure of data. A key limitation of the linear regression model is that it only captures average relationships in the data. It does not capture variations in the relationship between variables across areas or groups. Another key limitation is that the linear regression model can capture associations at either macro or micro levels, but it does not simultaneously measure their interdependencies.

To illustrate this, let us consider the regression intercept. It indicates that the average percentage of unemployed population at the OA level is 0.12 but this model ignores any spatial clustering i.e. the percentage of unemployed population tends to be similar across OAs nested within a same LSOA or MSOA. A side effect of ignoring this is that our standard errors are biased, and thus claims about statistical significance based on them would be misleading. Additionally, this situation also means we cannot explore variations in the percentage of unemployed population across LSOAs or MSOAs i.e. how the percentage of unemployed population may be dependent on various contextual factors at these geographical scales.

7.3.1.2 Fixed Effect Approach

An alternative approach is to adopt a fixed effects approach, or no-pooling model; that is, adding dummy variables indicating the group classification into the regression model e.g. the way OAs is nested within LSOAs (or MSOAs). This approach has limitations. First, there is high risk of overfitting. The number of groups may be too large, relative to the number of observations. Second, the estimation of multiple parameters may be required so that measuring differences between groups may be challenging. Third, a fixed effects approach does not allow including group-level explanatory variables. You can try fitting a linear regression model extending our estimated model to include dummy variables for individual LSOAs (and/or MSOAs) so you can compare this to the multilevel model below.

An alternative is fitting separate linear regression models for each group. This approach is not always possible if there are groups with small sizes.

7.4 Multilevel Modelling: Random Intercept Model

We use multilevel modelling to account for the hierarchical nature of the data by explicitly recognising that OAs are nested within LSOAs and MSOAs. Multilevel models can easily be estimated using in R using the package `lme4`. We implement an two-level model to allow for variation across LSOAs. We estimate an only intercept model allowing for variation across LSOAs. In essence, we are
estimating a model with varying intercept coefficient by LSOA. As you can see in the code chunk below, the equation has an additional component. This is the group component or LSOA effect. The $(1 \mid lsoa\_cd)$ means that we are allowing the intercept, represented by 1, to vary by LSOA.

```r
# specify a model equation
eq2 <- unemp ~ 1 + (1 | lsoa\_cd)
model2 <- lmer(eq2, data = oa\_shp)

# estimates
summary(model2)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: unemp ~ 1 + (1 | lsoa\_cd)
##    Data: oa\_shp
## REML criterion at convergence: -4382.6
## Scaled residuals:
##     Min 1Q Median 3Q Max
##-2.8741 -0.5531 -0.1215 0.4055 5.8207
## Random effects:
## Groups   Name   Variance Std.Dev.
## lsoa\_cd (Intercept) 0.002701  0.05197
## Residual          0.002575  0.05074
## Number of obs: 1584, groups: lsoa\_cd, 298
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   0.11432   0.003277   34.89
```

We can estimate a three-level model by adding $(1 \mid msoa\_cd)$ to allow the intercept to also vary by MSOAs and account for the nesting structure of LSOAs within MSOAs.

```r
# specify a model equation
eq3 <- unemp ~ 1 + (1 | lsoa\_cd) + (1 | msoa\_cd)
model3 <- lmer(eq3, data = oa\_shp)

# estimates
summary(model3)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: unemp ~ 1 + (1 | lsoa\_cd) + (1 | msoa\_cd)
##    Data: oa\_shp
## REML criterion at convergence: -4529.3
```
7.4. MULTILEVEL MODELLING: RANDOM INTERCEPT MODEL

## Scaled residuals:
## Min 1Q Median 3Q Max
## -2.5624 -0.5728 -0.1029 0.4228 6.1363
##
## Random effects:
## Groups    Name   Variance  Std.Dev.
## lsoa_cd   (Intercept)  0.0007603 0.02757
## msoa_cd   (Intercept)  0.0020735 0.04554
## Residual                                         0.0025723 0.05072
## Number of obs: 1584, groups: lsoa_cd, 298; msoa_cd, 61
##
## Fixed effects:
## Estimate Std. Error t value
## (Intercept) 0.115288 0.006187 18.64

We see two sets of coefficients: fixed effects and random effects. Fixed effects correspond to the standard linear regression coefficients. Their interpretation is as usual. Random effects are the novelty. It is a term in multilevel modelling and refers to varying coefficients i.e. the randomness in the probability of the model for the group-level coefficients. Specifically they relate to estimates of the average variance and standard deviation within groups (i.e. LSOAs or MSOAs). Intuitively, variance and standard deviation indicate the extent to which the intercept, on average, varies by LSOAs and MSOAs.

![Figure 7.2: Fig. 2. Variation of observations around their level 1 group mean.](image)

More formally, we first estimated the simplest regression model which is an intercept-only model and equivalent to the sample mean (i.e. the fixed part of the model):
Figure 7.3: Fig. 3. Variation of level 1 group mean around their level 2 group mean.

Figure 7.4: Fig. 4. Grand mean.
7.4. **Multilevel Modelling: Random Intercept Model**

\[ y_{ijk} = \mu + e_{ijk} \]

and then we made the random part of the model \( (e_{ijk}) \) more complex to account for the hierarchical structure of the data by estimating the following three-level regression model:

\[ y_{ijk} = \mu + u_{i..} + u_{ij..} + e_{ijk} \]

where \( y_{ijk} \) represents the proportion of unemployed population in OA \( i \) nested within LSOA \( j \) and MSOA \( k \); \( \mu \) represents the sample mean and the fixed part of the model; \( e_{ijk} \) is the deviation of an observation from its LSOA mean; \( u_{ij..} \) is the deviation of the LSOA mean from its MSOA mean; \( u_{i..} \) is the deviation of the MSOA mean from the fixed part of the model \( \mu \). Conceptually, this model is decomposing the variance of the model in terms of the hierarchical structure of the data. It is partitioning the observation’s residual into three parts or *variance components*. These components measure the relative extent of variation of each hierarchical level i.e. LSOA, MSOA and grand means. To estimate the set of residuals, they are assumed to follow a normal distribution and are obtained after fitting the model and are based on the estimates of the model parameters (i.e. intercept and variances of the random parameters).

Let’s now return to our three-level model (reported again below), we see that the intercept or fixed part of the model is the same as for the linear regression. The multilevel model reports greater standard errors. Multilevel models capture the hierarchical structure of the data and thus more precisely estimate the standard errors for our parameters.

```
# report model 3
summary(model3)
```

```r
## Linear mixed model fit by REML ['lmerMod']
## Formula: unemp ~ 1 + (1 | lsoa_cd) + (1 | msoa_cd)
## Data: oa_shp
##
## REML criterion at convergence: -4529.3
##
## Scaled residuals:
##    Min 1Q Median 3Q Max
## -2.5624 -0.5728 -0.1029 0.4228 6.1363
##
## Random effects:
## Groups   Name       Variance     Std.Dev.
## lsoa_cd  (Intercept) 0.0007603 0.02757
## msoa_cd  (Intercept) 0.0020735 0.04554
## Residual             0.0025723 0.05072
## Number of obs: 1584, groups:  lsoa_cd, 298; msoa_cd, 61
```
7.4.1 Interpretation

Fixed effects

We start by examining the fixed effects or estimated model averaging over LSOAs and MSOAs, $y_{ijk} = 0.115288$ which can also be called by executing:

```r
fixef(model3)
```

```
  (Intercept) 0.1152881
```

The estimated intercept indicates that the overall mean taken across LSOAs and MSOAs is estimated as $0.115288$ and is statistically significant at 5% significance.

Random effects

The set of random effects contains three estimates of variance and standard deviation and refer to the variance components discussed above. The `lsoa_cd`, `msoa_cd` and `Residual` estimates indicate that the extent of estimated LSOA-, MSOA- and individual-level variance is $0.0007603$, $0.0020735$ and $0.0025723$, respectively.

7.4.2 Variance Partition Coefficient (VPC)

The purpose of multilevel models is to partition variance in the outcome between the different groupings in the data. We thus often want to know the percentage of variation in the dependent variable accounted by differences across groups i.e. what proportion of the total variance is attributable to variation within-groups, or how much is found between-groups. The statistic to obtain this is termed the variance partition coefficient (VPC), or intra-class correlation. The VPC at the LSOA level indicates that 14% of the variation in percentage of unemployed resident population across OAs can be explained by differences across LSOAs. What is the VPC at the MSOA level?

```r
vpc_lsoa <- 0.0007603 / (0.0007603 + 0.0020735 + 0.0025723)
vpc_lsoa * 100
```

```
[1] 14.06374
```

The VPC is equal to the intra-class correlation coefficient which is the correlation between the observations of the dependent variable selected randomly from the same group. For instance, if the VPC is 0.1, we would say that 10% of the variation is between groups and 90% within. The correlation between randomly chosen pairs of observations belonging to the same group is 0.1.
You can also obtain the VPC by executing:

```r
#summ(model3)
```

### 7.4.3 Uncertainty of Estimates

You may have noticed that `lme4` does not provide p-values, because of various reasons as explained by Doug Bates, one of the author of `lme4`. These explanations mainly refer to the complexity of dealing with varying sample sizes at a given hierarchical level. The number of observations at each hierarchical level varies across individual groupings (i.e. LSOA or MSOA). It may even be one single observation. This has implications for the distributional assumptions, denominator degrees of freedom and how to approximate a “best” solution. Various approaches exist to compute the statistical significance of estimates. We use the `confint` function available within `lme4` to obtain confidence intervals.

```r
confint(model3, level = 0.95)
```

```
# Computing profile confidence intervals ...
# 2.5 %  97.5 %
# .sig01  0.02360251  0.03189046
# .sig02  0.03707707  0.05562307
# .sigma  0.04882281  0.05273830
# (Intercept) 0.10307341  0.12751103
```

`.sig01` refers to the LSOA level; `.sig02` refers to the MSOA level; and, `.sigma` refers to the OA level.

### 7.4.4 Assessing Group-level Variation

*Estimated regression coefficients*

In multilevel modelling, our primary interest is in knowing differences across groups. To visualise the estimated model within each group (i.e. LSOA and MSOA), we type:

```r
coeff_m3 <- coef(model3)
head(coeff_m3$lsoa_cd, 5)
```

```
 (Intercept)  
 E01006512 0.09915456
 E01006513 0.09889615
 E01006514 0.09297051
 E01006515 0.09803754
 E01006518 0.09642939
```

The results indicate that the estimated regression line is \( y = 0.09915456 \) for LSOA E01006512; \( y = 0.09889615 \) for LSOA E01006513 and so forth. Try getting the estimated model within each MSOA.
Random effects

We can look at the estimated group-level (or LSOA-level and MSOA-level) errors; that is, random effects:

```r
ranef_m3 <- ranef(model3)
head(ranef_m3$lsoa_cd, 5)
```

```
## (Intercept)
## E01006512 -0.01613353
## E01006513 -0.01639194
## E01006514 -0.02231758
## E01006515 -0.01725055
## E01006518 -0.01885870
```

Group-level errors indicate how much the intercept is shifted up or down in particular groups (i.e. LSOAs or MSOAs). Thus, for example, in LSOA E01006512, the estimated intercept is \(-0.01613353\) lower than average, so that the regression line is \((0.1152881 - 0.01613353) = 0.09915457\) which is what we observed from the call to `coef()`.

We can also obtain group-level errors (random effects) by using a simulation approach, labelled “Empirical Bayes” and discussed here. To this end, we run:

```r
# obtain estimates
REsim(model3) %>% head(10)
```

```
## groupFctr groupID term mean median sd
## 1 lsoa_cd E01006512 (Intercept) -0.017571250 -0.018409611 0.019856203
## 2 lsoa_cd E01006513 (Intercept) -0.017997503 -0.018431958 0.020338964
## 3 lsoa_cd E01006514 (Intercept) -0.023025554 -0.021409016 0.018419526
## 4 lsoa_cd E01006515 (Intercept) -0.014809761 -0.015182084 0.017941842
## 5 lsoa_cd E01006518 (Intercept) -0.017966169 -0.019754684 0.020449175
## 6 lsoa_cd E01006519 (Intercept) -0.016690370 -0.016403614 0.009121549
## 7 lsoa_cd E01006520 (Intercept) -0.024490783 -0.024416801 0.019491994
## 8 lsoa_cd E01006521 (Intercept) 0.005153600 0.005856537 0.018813114
## 9 lsoa_cd E01006522 (Intercept) 0.015744589 0.015072998 0.019172764
## 10 lsoa_cd E01006523 (Intercept) 0.002413828 0.002205473 0.018334849
```

The results contain the estimated mean, median and standard deviation for the intercept within each group (e.g. LSOA). The mean estimates are similar to those obtained from `ranef` with some small differences due to rounding.

To gain an understanding of the general pattern of the random effects, we can use caterpillar plots via `plotREsim` - reported below. The plot on the right shows the estimated random effects for each MSOA and their respective interval estimate. Note that random effects are on average zero, represented by the red horizontal line. Intervals that do not include zero are in bold. Also note that the width of the confidence interval depends on the standard error of the respective residual estimate, which is inversely related to the size of the sample. The residuals
represent an observation departures from the grand mean, so an observation
whose confidence interval does not overlap the line at zero (representing the
mean proportion of unemployed population across all areas) is said to differ
significantly from the average at the 5% level.

```r
# plot
plotREsim(REsim(model3))
```

Focusing on the plot on the right, we see MSOAs whose mean proportion of un-
employed population, assuming no explanatory variables, is lower than average.
On the right-hand side of the plot, you will see MSOAs whose mean proportion
is higher than average. The MSOAs with the smallest residuals include the dis-
tricts of Allerton and Hunt Cross, Church, Childwall, Wavertree and Woolton.
What districts do we have at the other extreme?

```r
re <- REsim(model3)
oa_shp %>% dplyr::select(msoa_cd, ward_nm, unemp) %>%
  filter(as.character(msoa_cd) == "E02001387" | as.character(msoa_cd) == "E02001393")
```

## Simple feature collection with 49 features and 3 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox: xmin: 339178.6 ymin: 386244.2 xmax: 341959.9 ymax: 389646.7
## projected CRS: Transverse_Mercator
## First 10 features:
## msoa_cd ward_nm unemp geometry
## 1 E02001393 Allerton and Hunts Cross 0.03246753 MULTIPOLYGON (((341333.6 38...
We can also map the MSOA-level random effects. To this end, we first need to read a shapefile containing data at the MSOA level and merge it with the random effects estimates.

```r
# read data
msoa_shp <- st_read("data/mlm/MSOA.shp")
```

```r
## Reading layer `MSOA' from data source `/home/jovyan/work/data/mlm/MSOA.shp' using driver `ESRI Shapefile'
## Simple feature collection with 61 features and 17 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox: xmin: 333086.1 ymin: 381426.3 xmax: 345636 ymax: 397980.1
## projected CRS: Transverse_Mercator
```

```r
# create a dataframe for MSOA-level random effects
re_msoa <- re %>% filter(groupFctr == "msoa_cd")
str(re_msoa)
```

```r
### 'data.frame': 61 obs. of 6 variables:
### $ groupFctr : chr "msoa_cd" "msoa_cd" "msoa_cd" "msoa_cd" ...
### $ groupID : chr "E02001347" "E02001348" "E02001349" "E02001350" ...
### $ term : chr "(Intercept)" "(Intercept)" "(Intercept)" "(Intercept)" ...
### $ mean : num -0.01156 -0.02505 -0.03463 0.00105 0.02509 ...
### $ median : num -0.011923 -0.025402 -0.035942 0.000172 0.025215 ...
### $ sd : num 0.0318 0.0294 0.0297 0.0314 0.016 ...
```

```r
# merge data
msoa_shp <- merge(x = msoa_shp, y = re_msoa, by.x = "MSOA_CD", by.y = "groupID")
```

Now we can create our map:

```r
# ensure geometry is valid
msoa_shp = sf::st_make_valid(msoa_shp)
```

```r
# create a map
legend_title = expression("MSOA-level residuals")
map_msoa = tm_shape(msoa_shp) +
  tm_fill(col = "mean", title = legend_title, palette = magma(256, begin = 0, end = 1)) +
  tm_borders(col = "white", lwd = .01)
```
7.4.5 Adding Individual-level Predictors

In this example, $\mu$ represents the sample mean but it could include a collection of independent variables or predictors. To explain the logic, we will assume that unemployment is strongly associated to long-term illness. We could expect that long-term illness ($lt_{ill}$) will reduce the chances of working and therefore being unemployed. Note that our focus is on the relationship, not on establishing causation. Specifically we want to estimate the relationship between unemployment and long-term illness and we are interested in variations in OA-level unemployment by MSOAs so we will estimate the following two-level model:

OA-level:

$$y_{ij} = \beta_{0j} + \beta_{1}x_{ij} + e_{ij}$$

MSOA-level:

$$\beta_{0j} = \beta_0 + u_{0j}$$

Replacing the first equation into the second, we have:
CHAPTER 7. MULTILEVEL MODELLING - PART 1

\[ y_{ij} = (\beta_0 + u_{0j}) + \beta_1 x_{ij} + e_{ij} \]

where \( y \) the proportion of unemployed population in OA \( i \) within MSOA \( j \); \( \beta_0 \) is the fixed intercept (averaging over all MSOAs); \( u_{0j} \) represents the MSOA-level residuals or random effects; \( \beta_0 \) and \( u_{0j} \) together represent the varying-intercept; \( \beta_1 \) is the slope coefficient; \( x_{ij} \) represents the percentage of long-term illness population; and, \( e_{ij} \) is the individual-level residuals.

We estimate the model executing:

```r
# change to proportion
oa_shp$lt_ill <- lt_ill/100

# specify a model equation
eq4 <- unemp ~ lt_ill + (1 | msoa_cd)
model4 <- lmer(eq4, data = oa_shp)

# estimates
summary(model4)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: unemp ~ lt_ill + (1 | msoa_cd)
## Data: oa_shp
##
## REML criterion at convergence: -4711.9
##
## Scaled residuals:
##    Min  1Q Median  3Q Max
## -5.1941 -0.5718 -0.0906 0.4507 5.9393
##
## Random effects:
##  Groups     Name     Variance Std.Dev.
##  msoa_cd   (Intercept) 0.001421 0.03769
##       Residual      0.002674 0.05171
## Number of obs: 1584, groups: msoa_cd, 61
##
## Fixed effects:
##                Estimate Std. Error t value
## (Intercept) 0.04682     0.00625   7.492
## lt_ill     0.29588     0.01615  18.317
##
## Correlation of Fixed Effects:
##   (Intr)
## lt_ill -0.600
```

*Fixed effects:* model averaging over MSOAs
7.4. MULTILEVEL MODELLING: RANDOM INTERCEPT MODEL

```r
fixef(model4)
```

```r
## (Intercept)  lt_ill
## 0.04681959  0.29588110
```
yields an estimated regression line in an average McSOA: 
\[ y = 0.04681959 + 0.29588110x \]

Random effects: MSOA-level errors

```r
ranef_m4 <- ranef(model4)
head(ranef_m4$msoa_cd, 5)
```

```r
## (Intercept) 
## E02001347 -0.017474815 
## E02001348 -0.021203807 
## E02001349 -0.022469313 
## E02001350 -0.003539869 
## E02001351 0.008502813 
```
yields an estimated intercept for MSOA E02001347 which is 0.017474815 lower than the average with a regression line: 
\[ (0.04681959 - 0.017474815) + 0.29588110x = 0.02934478 + 0.29588110x \].

You can confirm this by looking at the estimated model within each MSOA by executing (remove the # sign):

```r
#coef(model4)
```

Fixed effect correlations

In the bottom of the output, we have the correlations between the fixed-effects estimates. In our example, it refers to the correlation between \( \beta_0 \) and \( \beta_1 \). It is negative indicating that in MSOAs where the relationship between unemployment and long-term illness is greater, as measured by \( \beta_1 \), the average proportion of unemployed people tends to be smaller, as captured by \( \beta_0 \).

7.4.6 Adding Group-level Predictors

We can also add group-level predictors. We use the formulation:

OA-level:

\[ y_{ij} = \beta_{0j} + \beta_1 x_{ij} + e_{ij} \]

MSOA-level:

\[ \beta_{0j} = \beta_0 + \gamma_1 m_j + u_{0j} \]
where $x_{ij}$ is the OA-level proportion of population suffering long-term illness and $m_j$ is the MSOA-level proportion of male population. We first need to create this group-level predictor:

```r
# detach OA shp and attach MSOA shp
detach(oa_shp)
attach(msoa_shp)

# group-level predictor
msoa_shp$pr_male <- males/pop

# remove geometries
msoa_df <- `st_geometry<-(msoa_shp, NULL)

# select variables
msoa_df <- msoa_df %>% dplyr::select(MSOA_CD, pop, pr_male)

# merge data sets
oa_shp <- merge(x=oa_shp, y=msoa_df, by.x = "msoa_cd", by.y="MSOA_CD")

# inspect data
head(oa_shp[1:10, c("msoa_cd", "oa_cd", "unemp", "pr_male")])
```

We can now estimate our model:

```r
detach(msoa_shp)
attach(oa_shp)

# specify a model equation
eq5 <- unemp ~ lt_ill + pr_male + (1 | msoa_cd)
model5 <- lmer(eq5, data = oa_shp)

# estimates
summary(model5)
```
7.4. MULTILEVEL MODELLING: RANDOM INTERCEPT MODEL

## Linear mixed model fit by REML ['lmerMod']
## Formula: unemp ~ lt_ill + pr_male + (1 | msoa_cd)
## Data: oa_shp
##
## REML criterion at convergence: -4712.3
##
## Scaled residuals:
## Min 1Q Median 3Q Max
## -5.2162 -0.5696 -0.0929 0.4549 5.9370
##
## Random effects:
## Groups Name Variance Std.Dev.
## msoa_cd (Intercept) 0.001391 0.03729
## Residual 0.002674 0.05171
## Number of obs: 1584, groups: msoa_cd, 61
##
## Fixed effects:
## Estimate Std. Error t value
## (Intercept) -0.07746 0.08768 -0.883
## lt_ill 0.29781 0.01620 18.389
## pr_male 0.25059 0.17642 1.420
##
## Correlation of Fixed Effects:
## (Intr) lt_ill
## lt_ill -0.118
## pr_male -0.997 0.075

This model includes the proportion of males and intercepts that vary by MSOA. The \( \text{lmer()} \) function only accepts predictors at the individual level, so we have included data on the proportion of male population at this level. Explore and interpret the model running the functions below:

```r
# fixed effects
fixef(model5)

# random effects
ranef_m5 <- ranef(model5)
head(ranef_m5$msoa_cd, 5)
```

```r
# (Intercept) lt_ill pr_male
# -0.0774607 0.2978084 0.2505913
```

```r
# (Intercept)
# E02001347 -0.013625261
# E02001348 -0.019757846
# E02001349 -0.023709992
# E02001350 0.003003861
# E02001351 0.003508477
```
Adding group-level predictors tends to improve inferences for group coefficients. Examine the confidence intervals, in order to evaluate how the precision of our estimates of the MSOA intercepts have changed. Have confidence intervals for the intercepts of Model 4 and 5 increased or reduced? Hint: look at how to get the confidence intervals above.

## 7.5 Questions

For the second assignment, we will be using a different dataset comprising information on COVID-19 cases, census data and the Index of Multiple Deprivation (IMD) for England. The data set is similar in structured to that used in this chapter. It is hierarchically organised into 149 Upper Tier Local Authority Districts (UTLADs) within 9 Regions and has 508 variables - see Chapter 11 for a more detailed description of the data.

```r
sdf <- st_read("data/assignment_2_covid/covid19_eng.gpkg")
```

Here we see a selection of 10 variables for 5 UTLADs.

```r
head(sdf[1:5,c(3,4,9,10,381,385,386,387,403,406)])
```

## 7.5 Questions

For the second assignment, we will be using a different dataset comprising information on COVID-19 cases, census data and the Index of Multiple Deprivation (IMD) for England. The data set is similar in structured to that used in this chapter. It is hierarchically organised into 149 Upper Tier Local Authority Districts (UTLADs) within 9 Regions and has 508 variables - see Chapter 11 for a more detailed description of the data.

```r
sdf <- st_read("data/assignment_2_covid/covid19_eng.gpkg")
```

Here we see a selection of 10 variables for 5 UTLADs.

```r
head(sdf[1:5,c(3,4,9,10,381,385,386,387,403,406)])
```
7.5. QUESTIONS

## MULTIPOLYGON (((449862.8 52...
## MULTIPOLYGON (((455939.7 52...
## MULTIPOLYGON (((444126.1 52...
## MULTIPOLYGON (((423475.7 52...

- ctyual9nm: Upper Tier Local Authority District name
- Region: Region name
- X2020.01.31: COVID-19 cases on January 31st 2020
- X2020.02.01: COVID-19 cases on February 1st 2020
- IMD...Average.score: Average IMD score for UTLADs - see File 11: upper-tier local authority summaries for information on this and associated indicators.
- Residents: Number of residents
- Households: Number of households
- Dwellings: Number of dwellings
- Age_85plus: Number of people aged 85 and over
- White_British_and_Irish: Number of white British and Irish people

Note that variable names relating to the daily COVID-19 cases are organised in the following way: X stands for daily COVID-19 cases, followed by the year (i.e. 2020, 2021); month (i.e. January to December); and day (i.e. 01 to 31).

Using these data, you are required to address the following challenges:

1. Fit a varying-intercept model with no explanatory variables. Let the intercept to vary by region.
2. Fit a varying-intercept model with including at least three explanatory variables.
3. Compute the Variance Partition Coefficient (VPC) for the models estimated according to points 1 and 2 above.
4. Create caterpillar plots to visualise the varying intercepts.

Analyse and discuss: 1. the extent of variation in the dependent variables at the two geographical scales (variation at which geographical scale explains most of variance in your dependent variable); 2. the varying intercept estimate(s) from your model(s) (what can they tell you about the difference between groups / areas? are they statistically significantly different?);

Ensure you appropriately describe the structure of the data and identify the various geographical scales of analysis (i.e. level-1 and level-2 units)

In addressing the challenges in this and following chapters, you have some flexibility to be creative. A set of key factors to consider:

1. **Dependent Variable**: We will seek to explain daily COVID-19 cases, and you will need to make a decision as to:
   - **Daily vs cumulative COVID-19 cases**. Given that we will be focusing on cross-sectional models (i.e. models for one snapshot), you can focus on
modelling daily cases at one specific date or cumulative daily cases over a period of time.

- **Time period.** You can select the date or period of time that you will focus your analysis on.

- **Use risk of COVID-19 infection.** The dependent variable should be the risk or rate of COVID-19 infection.

For example, the risk of COVID-19 infection for the period (i.e. between Dec. 1st, 2020 - January 29th, 2021) comprising the third wave of the pandemic in the United Kingdom can be computed as:

```r
# computing cumulative COVID cases for 01/12/2020 - 29/01/2021
sdf[, 509] <- sdf %>% dplyr::select(!X2020.12.01:X2021.01.29) %>% # select COVID cases
             mutate(cum_covid = rowSums(across(where(is.numeric)))) %>% # sum daily cases
             dplyr::select(cum_covid) %>% # select cumulative cases
             st_set_geometry(., NULL) # set geometry to NULL

# computing risk of infection
sdf <- sdf %>% mutate(
              covid19_r = round((cum_covid / Residents) * 1000)
)
```

2. **Explanatory variables:**

- **At least 3.** Use at least 3 explanatory variables. There is no maximum limit but consider your model to be parsimonious.

- **Choice your set.** Select the set of variables you consider appropriate / interesting. Make sure you justify your choice based on evidence and/or theory.

- **Percentages / Proportions.** Use percentages or proportions to capture the composition of places, rather than numbers of people, households or dwellings. For this, ensure you are using the appropriate denominator.

For instance, if you want to capture the relationship between cumulative COVID-19 cases and overcrowding, share of elderly population and nonwhite minorities, use the following variables:

```r
sdf <- sdf %>% mutate(
  crowded_hou = Crowded_housing / Households, # share of crowded housing
  elderly = (Age_85plus) / Residents, # share of population aged 65+
  ethnic = (Mixed + Indian + Pakistani + Bangladeshi + Chinese + Other_Asian + Black +
)
```

**ADVICE:** Create a new spatial data frame including only the variables you will analyse. For example:
nsdf <- sdf %>% dplyr::select(objectid, ctyua19cd, ctyua19nm, Region, covid19_r, crowded_hou, elderly, ethnic, Residents)
Chapter 8

Multilevel Modelling - Part 2

This chapter explains varying slopes and draws on the following references:

The content of this chapter is based on:

- Gelman and Hill (2006b) provides an excellent and intuitive explanation of multilevel modelling and data analysis in general. Read Part 2A for a really good explanation of multilevel models.
- for Multilevel Modelling (nd) is an useful online resource on multilevel modelling and is free!

8.1 Dependencies

This chapter uses the following libraries which are listed in the Dependency list Section of Chapter 1:

```r
# Data manipulation, transformation and visualisation
library(tidyverse)

## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1

# Nice tables
library(kableExtra)

# Simple features (a standardised way to encode vector data ie. points, lines, polygons)
library(sf)

# Spatial objects conversion
library(sp)

# Thematic maps
```

147
8.2 Data

For this chapter, we will data for Liverpool from England’s 2011 Census. The original source is the Office of National Statistics and the dataset comprises a number of selected variables capturing demographic, health and socio-economic of the local resident population at four geographic levels: Output Area (OA), Lower Super Output Area (LSOA), Middle Super Output Area (MSOA) and Local Authority District (LAD). The variables include population counts and percentages. For a description of the variables, see the readme file in the mlm data folder.

Let us read the data:

```r
# clean workspace
rm(list=ls())
# read data
oa_shp <- st_read("data/mlm/OA.shp")
```

8.3 Conceptual Overview

So far, we have estimated varying-intercept models; that is, when the intercept ($\beta_0$) is allowed to vary by group (eg. geographical area) - as shown in Fig. 1(a). The strength of the relationship between $y$ (i.e. unemployment rate) and $x$ (long-term illness) has been assumed to be the same across groups (i.e. MSOAs), as captured by the regression slope ($\beta_1$). Yet it can also vary by group as shown in Fig. 1(b), or we can observe group variability for both intercepts and slopes as represented in Fig. 1(c).

---

1Read the file in R by executing `read_tsv("data/mlm/readme.txt")`
8.3. CONCEPTUAL OVERVIEW

Figure 8.1: Fig. 1. Linear regression model with (a) varying intercepts, (b) varying slopes, and (c) both. Source: Gelman and Hill (2006b) p.238.

8.3.1 Exploratory Analysis: Varying Slopes

Let’s then explore if there is variation in the relationship between unemployment rate and the share of population in long-term illness. We do this by selecting the 8 MSOAs containing OAs with the highest unemployment rates in Liverpool.

```r
# Sort data
oa_shp <- oa_shp %>% arrange(-unemp)
oa_shp[1:9, c("msoa_cd", "unemp")]
```

---

```r
# Simple feature collection with 9 features and 2 fields
# geometry type: MULTIPOLYGON
# dimension: XY
# bbox: xmin: 335032 ymin: 387777 xmax: 338576.1 ymax: 395022.4
# projected CRS: Transverse_Mercator
# msoa_cd unemp geometry
## 1 E02001354 0.500000 MULTIPOLYGON (((337491.2 39...  
## 2 E02001369 0.496063 MULTIPOLYGON (((335272.3 39...  
## 3 E02001366 0.446154 MULTIPOLYGON (((338198.1 39...  
## 4 E02001365 0.435294 MULTIPOLYGON (((336572.2 39...  
## 5 E02001370 0.402439 MULTIPOLYGON (((336328.3 39...  
## 6 E02001390 0.380166 MULTIPOLYGON (((335833.6 38...  
## 7 E02001354 0.375000 MULTIPOLYGON (((337403 3949...  
## 8 E02001385 0.370787 MULTIPOLYGON (((336251.6 38...  
## 9 E02001368 0.364865 MULTIPOLYGON (((335209.3 39...  

# Select MSOAs
s_t8 <- oa_shp %>% dplyr::filter(
  as.character(msoa_cd) %in% c("E02001354", 
  "E02001369", 
  "E02001366", 
  "E02001365", 
```

---
And then we generate a set of scatter plots and draw regression lines for each MSOA.

```r
ggplot(s_t8, aes(x = lt_ill, y = unemp)) +
  geom_point() +
  geom_smooth(method = "lm") +
  facet_wrap(~ msoa_cd, nrow = 2) +
  ylab("Unemployment rate") +
  xlab("Long-term Illness (%)") +
  theme_classic()
```

## `geom_smooth()` using formula 'y ~ x'

We can observe great variability in the relationship between unemployment rates and the percentage of population in long-term illness. A strong and positive relationship exists in MSOA E02001366 (Tuebrook and Stoneycroft), while it is negative in MSOA E02001370 (Everton) and neutral in MSOA E02001390 (Princes Park & Riverside). This visual inspection suggests that accounting for differences in the way unemployment rates relate to long-term illness is important. Contextual factors may differ across MSOAs in systematic ways.
8.4 Estimating Varying Intercept and Slopes Models

A way to capture for these group differences in the relationship between unemployment rates and long-term illness is to allow the relevant slope to vary by group (i.e. MSOA). We can do this estimating the following model:

**OA-level:**

\[ y_{ij} = \beta_{0j} + \beta_{1j}x_{ij} + e_{ij} \]

**MSOA-level:**

\[ \beta_{0j} = \beta_0 + u_{0j} \]
\[ \beta_{1j} = \beta_1 + u_{1j} \]

Replacing the first equation into the second generates:

\[ y_{ij} = (\beta_0 + u_{0j}) + (\beta_1 + u_{1j})x_{ij} + e_{ij} \]

where, as in the previous Chapter, \( y \) the proportion of unemployed population in OA \( i \) within MSOA \( j \); \( \beta_0 \) is the fixed intercept (averaging over all MSOAs); \( u_{0j} \) represents the MSOA-level residuals, or *random effects*, of the intercept; \( e_{ij} \) is the individual-level residuals; and, \( x_{ij} \) represents the percentage of long-term illness population. *But* now we have a varying slope represented by \( \beta_1 \) and \( u_{1j} \):

- \( \beta_1 \) is estimated average slope - fixed part of the model; and,
- \( u_{1j} \) is the estimated group-level errors of the slope.

To estimate such model, we add `lt_ill` in the bracket with a `+` sign between 1 and | i.e. \((1 + lt_ill | msoa_cd)\).

```r
# attach df
attach(oa_shp)

# change to proportion
oa_shp$lt_ill <- lt_ill/100

# specify a model equation
eq6 <- unemp ~ lt_ill + (1 + lt_ill | msoa_cd)
model6 <- lmer(eq6, data = oa_shp)

# estimates
summary(model6)
```

## Linear mixed model fit by REML ['lmerMod']
## Formula: unemp ~ lt_ill + (1 + lt_ill | msoa_cd)
## Data: oa_shp

## REML criterion at convergence: -4762.8

## Scaled residuals:

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-3.66</td>
<td>-0.57</td>
<td>-0.09</td>
<td>0.46</td>
<td>5.49</td>
</tr>
</tbody>
</table>

## Random effects:

<table>
<thead>
<tr>
<th>Groups</th>
<th>Name</th>
<th>Variance</th>
<th>Std.Dev.</th>
<th>Corr</th>
</tr>
</thead>
<tbody>
<tr>
<td>msoa_cd</td>
<td>(Intercept)</td>
<td>0.003428</td>
<td>0.05855</td>
<td></td>
</tr>
<tr>
<td></td>
<td>lt_ill</td>
<td>0.029425</td>
<td>0.17154</td>
<td>-0.73</td>
</tr>
<tr>
<td></td>
<td>Residual</td>
<td>0.002474</td>
<td>0.04974</td>
<td></td>
</tr>
</tbody>
</table>

Number of obs: 1584, groups: msoa_cd, 61

## Fixed effects:

<table>
<thead>
<tr>
<th></th>
<th>Estimate</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.047650</td>
<td>0.008635</td>
<td>5.519</td>
</tr>
<tr>
<td>lt_ill</td>
<td>0.301259</td>
<td>0.028162</td>
<td>10.697</td>
</tr>
</tbody>
</table>

## Correlation of Fixed Effects:

<table>
<thead>
<tr>
<th></th>
<th>(Intr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lt_ill</td>
<td>-0.786</td>
</tr>
</tbody>
</table>

In this model, the estimated standard deviation of the unexplained within-MSOA variation is 0.04974, and the estimated standard deviation of the MSOA intercepts is 0.05855. But, additionally, we also have estimates of standard deviation of the MSOA slopes (0.17154) and correlation between MSOA-level residuals for the intercept and slope (-0.73). While the former measures the extent of average deviation in the slopes across MSOAs, the latter indicates that the intercept and slope MSOA-level residuals are negatively associated; that is, MSOAs with large slopes have relatively smaller intercepts and vice versa. We will come back to this in Section Interpreting Correlations Between Group-level Intercepts and Slopes.

Similarly, the correlation of fixed effects indicates a negative relationship between the intercept and slope of the average regression model; that is, as the average model intercept tends to increase, the average strength of the relationship between unemployment rate and long-term illness decreases and vice versa.

We then explore the estimated average coefficients (fixed effects):

```r
fixef(model6)
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.04765009</td>
</tr>
<tr>
<td>lt_ill</td>
<td>0.30125875</td>
</tr>
</tbody>
</table>

yields an estimated regression line in an average LSOA: \( y = 0.04764998 + \)
0.30125916. The fixed intercept indicates that the average unemployment rate is 0.05 if the percentage of population with long-term illness is zero. The fixed slope indicates that the average relationship between unemployment rate and long-term illness is positive across MSOAs i.e. as the percentage of population with long-term illness increases by 1 percentage point, the unemployment rate increases by 0.3.

We look the estimated MSOA-level errors (*random effects*):

```r
text <- ranef(model6)
head(text$msoa_cd, 5)
```

<table>
<thead>
<tr>
<th></th>
<th>(Intercept)</th>
<th>lt_ill</th>
</tr>
</thead>
<tbody>
<tr>
<td>E02001347</td>
<td>-0.026561345</td>
<td>0.02718102</td>
</tr>
<tr>
<td>E02001348</td>
<td>0.001688245</td>
<td>-0.11533102</td>
</tr>
<tr>
<td>E02001349</td>
<td>-0.036084817</td>
<td>0.05547075</td>
</tr>
<tr>
<td>E02001350</td>
<td>0.032240842</td>
<td>-0.14298734</td>
</tr>
<tr>
<td>E02001351</td>
<td>0.086214137</td>
<td>-0.28130162</td>
</tr>
</tbody>
</table>

Recall these estimates indicate the extent of deviation of the MSOA-specific intercept and slope from the estimated model average captured by the fixed model component.

We can also regain the estimated intercept and slope for each county by adding the estimated MSOA-level errors to the estimated average coefficients; or by executing:

```r
#coef(model6)
```

We are normally more interested in identifying the extent of deviation and its significance. To this end, we create a caterpillar plot:

```r
# plot
plotREsim(REsim(model6))
```
These plots reveal some interesting patterns. First, only one MSOA, containing wards such as Tuebrook and Stoneycroft, Anfield & Everton, seems to have a statistically significantly different intercept, or average unemployment rate. Confidence intervals overlap zero for all other 60 MSOAs. Despite this, note that when a slope is allowed to vary by group, it generally makes sense for the intercept to also vary. Second, significant variability exists in the association between unemployment rate and long-term illness across MSOAs. Ten MSOAs display a significant positive association, while 12 exhibit a significantly negative relationship. Third, these results reveal that geographical differences in the relationship between unemployment rate and long-term illness can explain the significant differences in average unemployment rates in the varying intercept only model.

Let’s try to get a better understanding of the varying relationship between unemployment rate and long-term illness by mapping the relevant MSOA-level errors.

```r
# read data
msoa_shp <- st_read("data/mlm/MSOA.shp")
```

---

**Number of features:**
- **61**

**Fields:**
- **17**

**Geometry type:** MULTIPOLE

**Dimension:** XY

**Bbox:**
- xmin: 333086.1
- ymin: 381426.3
- xmax: 345636
- ymax: 397980.1

**Projected CRS:** Transverse_Mercator
# create a dataframe for MSOA-level random effects
re_msoa_m6 <- REsim(model6) %>% filter(groupFctr == "msoa_cd") %>%
  filter(term == "lt_ill")
str(re_msoa_m6)

# merge data
msoa_shp <- merge(x = msoa_shp, y = re_msoa_m6, by.x = "MSOA_CD", by.y = "groupID")

# ensure geometry is valid
msoa_shp = sf::st_make_valid(msoa_shp)

# create a map
legend_title = expression("MSOA-level residuals")
map_msoa = tm_shape(msoa_shp) +
tm_fill(col = "median", title = legend_title, palette = magma(256, begin = 0, end = 1), style = "cont") +
tm_borders(col = "white", lwd = 0.01) +
tm_compass(type = "arrow", position = c("right", "top") , size = 4) +
tm_scale_bar(breaks = c(0,1,2), text.size = 0.5, position = c("center", "bottom"))
map_msoa
The map indicates that the relationship between unemployment rate and long-term illness is stronger and positive in northern MSOAs; that is, the percentage of population with long-term illness explains a greater share of the variation in unemployment rates in these locations. As expected, a greater share of population in long-term illness is associated with higher local unemployment. In contrast, the relationship between unemployment rate and long-term illness tends to operate in the reverse direction in north-east and middle-southern MSOAs. In these MSOAs, OAs tend to have a higher unemployment rate relative the share of population in long-term illness. You can confirm this examining the data for specific MSOA executing:

```r
oa_shp %>% dplyr::select(msoa_cd, ward_nm, unemp, lt_ill) %>%
  filter(as.character(msoa_cd) == "E02001370")
```

```
## Simple feature collection with 23 features and 4 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox:   xmin: 335885 ymin: 391134.2 xmax: 337596.3 ymax: 392467
## projected CRS: Transverse_Mercator
## First 10 features:
## msoa_cd  ward_nm  unemp  lt_ill
## 1 E02001370  Everton  0.4024390  0.2792793
## 2 E02001370 Tuebrook and Stoneycroft  0.3561644  0.3391813
## 3 E02001370  Everton  0.3285714  0.3106383
## 4 E02001370  Everton  0.3209877  0.3283019
## 5 E02001370     Anfield  0.3082707  0.1785714
```
8.5. INTERPRETING CORRELATIONS BETWEEN GROUP-LEVEL INTERCEPTS AND SLOPES

Correlations of random effects are confusing to interpret. Key for their appropriate interpretation is to recall they refer to group-level residuals i.e. deviation of intercepts and slopes from the average model intercept and slope. A strong negative correlation indicates that groups with high intercepts have relatively low slopes, and vice versa. A strong positive correlation indicates that groups with high intercepts have relatively high slopes, and vice versa. A correlation close to zero indicate little or no systematic between intercepts and slopes. Note that a high correlation between intercepts and slopes is not a problem, but it makes the interpretation of the estimated intercepts more challenging. For this reason, a suggestion is to center predictors \( (x') \); that is, subtract their average value \( (z = x - \bar{x}) \). For a more detailed discussion, see for Multilevel Modelling (nd).

To illustrate this, let’s reestimate our model adding an individual-level predictor: the share of population with no educational qualification.

```r
# centering to the mean
oa_shp$z_no_qual <- no_qual/100 - mean(no_qual/100)
oa_shp$z_lt_ill <- lt_ill - mean(lt_ill)

# specify a model equation
eq7 <- unemp ~ z_lt_ill + z_no_qual + (1 + z_lt_ill | msoa_cd)
```
model7 <- lmer(eq7, data = oa_shp)

# estimates
summary(model7)

## Linear mixed model fit by REML ['lmerMod']
## Formula: unemp ~ z_lt_ill + z_no_qual + (1 + z_lt_ill | msoa_cd)
## Data: oa_shp
##
## REML criterion at convergence: -4940.7
##
## Scaled residuals:
## Min 1Q Median 3Q Max
## -3.6830 -0.5949 -0.0868 0.4631 6.3556
##
## Random effects:
## Groups Name Variance Std.Dev. Corr
## msoa_cd (Intercept) 8.200e-04 0.02864
## z_lt_ill 2.161e-06 0.00147 -0.04
## Residual 2.246e-03 0.04739
## Number of obs: 1584, groups: msoa_cd, 61
##
## Fixed effects:
## Estimate Std. Error t value
## (Intercept) 0.1163682 0.0039201 29.68
## z_lt_ill -0.0003130 0.0003404 -0.92
## z_no_qual 0.3245811 0.0221347 14.66
##
## Correlation of Fixed Effects:
## (Intr) z_lt_l
## z_lt_ill -0.007
## z_no_qual -0.015 -0.679

How do you interpret the random effect correlation?

8.6 Model building

Now we know how to estimate multilevel regression models in R. The question that remains is: When does multilevel modeling make a difference? The short answer is: when there is little group-level variation. When there is very little group-level variation, the multilevel modelling reduces to classical linear regression estimates with no group indicators. Inversely, when group-level coefficients vary greatly (compared to their standard errors of estimation), multilevel modelling reduces to classical regression with group indicators Gelman and Hill (2006b).
8.7. QUESTIONS

How do you go about building a model?

We generally start simple by fitting simple linear regressions and then work our way up to a full multilevel model - see Gelman and Hill (2006b) p. 270.

How many groups are needed?

As an absolute minimum, more than two groups are required. With only one or two groups, a multilevel model reduces to a linear regression model.

How many observations per group?

Two observations per group is sufficient to fit a multilevel model.

8.6.1 Model Comparison

How we assess different candidate models? We can use the function `anova()` and assess various statistics: The Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), Loglik and Deviance. Generally, we look for lower scores for all these indicators. We can also refer to the $\text{Chisq}$ statistic below. It tests the hypothesis of whether additional predictors improve model fit. Particularly it tests the Null Hypothesis whether the coefficients of the additional predictors equal 0. It does so comparing the deviance statistic and determining if changes in the deviance are statistically significant. Note that a major limitation of the deviance test is that it is for nested models i.e. a model being compared must be nested in the other. Below we compare our two models. The results indicate that adding an individual-level predictor (i.e. the share of population with no qualification) provides a model with better.

```r
anova(model6, model7)
```

## refitting model(s) with ML (instead of REML)

## Data: oa_shp

## Models:

## model6: unemp ~ lt_ill + (1 + lt_ill | msoa_cd)
## model7: unemp ~ z_lt_ill + z_no_qual + (1 + z_lt_ill | msoa_cd)

## npar AIC BIC logLik deviance Chisq Df Pr(>Chisq)
## model6 6 -4764.7 -4732.5 2388.3 -4776.7
## model7 7 -4956.5 -4918.9 2485.2 -4970.5 193.76 1 < 2.2e-16 ***

## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

8.7 Questions

We will continue to use the COVID-19 dataset. Please see Chapter 11 for details on the data.
sdf <- st_read("data/assignment_2_covid/covid19_eng.gpkg")

## Reading layer `covid19_eng' from data source `/home/jovyan/work/data/assignment_2_covid/covid19_eng.gpkg' using driver `GPKG'
## Simple feature collection with 149 features and 507 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox: xmin: 134112.4 ymin: 11429.67 xmax: 655653.8 ymax: 657536
## projected CRS: OSGB 1936 / British National Grid

Using these data, you are required to address the following challenges:

1. Fit a varying-slope model. Let one slope to vary by region. Think carefully your choice.
2. Fit a varying-intercept and varying-slope model.
3. Compare the results for models fitted in 1 and 2. Which is better? Why?

Use the same explanatory variables used for the Chapter 7 challenge, so you can compare the model results from this chapter.

Analyse and discuss:

1. the varying slope estimate(s) from your model(s) (to what extent does the relationship between your dependent and independent variables vary across groups / areas? are they statistically significantly different?).
2. differences between your varying intercept and varying slope models.
Chapter 9

Geographically Weighted Regression

This chapter provides an introduction to geographically weighted regression models.

The content of this chapter is based on:

- Fotheringham et al. (2002), a must-go book if you are working or planning to start working on geographically weighted regression modelling.
- Comber et al. (2020)’s recently published preprint which provides a roadmap to approach various practical issues in the application of GWR.

9.1 Dependencies

This chapter uses the following libraries:

```r
# Data manipulation, transformation and visualisation
library(tidyverse)

## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1

# Nice tables
library(kableExtra)

# Simple features (a standardised way to encode vector data ie. points, lines, polygons)
library(sf)

# Spatial objects conversion
library(sp)

# Thematic maps
library(tmap)
```
9.2 Data

For this chapter, we will use data on:

- cumulative COVID-19 confirmed cases from 1st January, 2020 to 14th April, 2020 from Public Health England via the GOV.UK dashboard;

- resident population characteristics from the 2011 census, available from the Office of National Statistics; and,

- 2019 Index of Multiple Deprivation (IMD) data from GOV.UK and published by the Ministry of Housing, Communities & Local Government.

The data used for this Chapter are organised at the ONS Upper Tier Local Authority (UTLA) level - also known as Counties and Unitary Authorities. They are the geographical units used to report COVID-19 data.

If you use the dataset utilised in this chapter, make sure cite this book. For a full list of the variables included in the data set used in this Chapter, see the readme file in the gwr data folder.¹

Let’s read the data:

```r
# clean workspace
rm(list=ls())
# read data
utla_shp <- st_read("data/gwr/Covid19_total_cases_geo.shp") %>%
  select(objct, cty19c, ctyu19nm, long, lat, st_rs, st_ln, X2020.04.14, I.PL1, IMD20, IMD2., Rsdnt, Hshld, Dwlln, Hsh_S, E_16_, A_65_, Ag_85, Mixed, Indin, Pkstn, Bngld, Chins, Oth_A, Black, Othr_t, CB_U_, Crwd_, Lng__, Trn__, Adm__, Ac___, Pb___, Edctn, H____, geometry)
# replace nas with 0s
utla_shp[is.na(utla_shp)] <- 0
```

¹Read the file in R by executing `read_tsv("data/gwr/readme.txt")`
# Recap: Spatial Effects

To this point, we have implicitly discussed three distinctive spatial effects:

- **Spatial heterogeneity** refers to the uneven distribution of a variable’s values across space.

- **Spatial dependence** refers to the spatial relationship of a variable’s values for a pair of locations at a certain distance apart, so that they are more similar (or less similar) than expected for randomly associated pairs of observations.

- **Spatial nonstationarity** refers to variations in the relationship between an outcome variable and a set of predictor variables across space.

In previous sessions, we considered multilevel models to deal with spatial nonstationarity, recognising that the strength and direction of the relationship between an outcome $y$ and a set of predictors $x$ may vary over space. Here we consider a different approach, namely geographically weighted regression (GWR).

## Exploratory Analysis

We will explore this technique through an empirical analysis considering the current global COVID-19 outbreak. Specifically we will seek to identify potential contextual factors that may be related to an increased risk of local infection. Population density, overcrowded housing, vulnerable individuals and critical workers have all been linked to a higher risk of COVID-19 infection.

First, we will define and develop some basic understanding of our variable of interest. We define the risk of COVID-19 infection by the cumulative number of confirmed positive cases COVID-19 per 100,000 people:

```r
# risk of covid-19 infection
utla_shp$covid19_r <- (utla_shp$X2020.04.14 / utla_shp$Rsdnt) * 100000
```

```r
# histogram
ggplot(data = utla_shp) + geom_density(alpha=0.8, colour="black", fill="lightblue", aes(x = covid19_r)) + theme_classic()
```
The results indicate a wide variation in the risk of infection across UTLAs in England, ranging from 31 to 342 confirmed positive cases of COVID-19 per 100,000 people with a median of 147. We map the cases to understand their spatial structure.

```r
# distribution in numbers
summary(utla_shp$covid19_r)
```

```
##    Min. 1st Qu.  Median     Mean 3rd Qu.     Max.     NA's
##  32.11  92.81  140.15  146.66 190.96  341.56      0
```

The results indicate a wide variation in the risk of infection across UTLAs in England, ranging from 31 to 342 confirmed positive cases of COVID-19 per 100,000 people with a median of 147. We map the cases to understand their spatial structure.

```r
# read region boundaries for a better looking map
reg_shp <- st_read("data/gwr/Regions_December_2019_Boundaries_EN_BGC.shp")
```

```r
## Reading layer `Regions_December_2019_Boundaries_EN_BGC' from data source `/home/jovyan/work/data/gwr/Regions_December_2019_Boundaries_EN_BGC.shp' using driver `ESRI Shapefile'
## Simple feature collection with 9 features and 9 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox: xmin: 82672 ymin: 5342.7 xmax: 655653.8 ymax: 657536
## projected CRS: OSGB 1936 / British National Grid
```

```r
# ensure geometry is valid
utla_shp = sf::st_make_valid(utla_shp)
reg_shp = sf::st_make_valid(reg_shp)
```

```r
# map
legend_title = expression("Cumulative cases per 100,000")
```
The map shows that concentrations of high incidence of infections in the metropolitan areas of London, Liverpool, Newcastle, Sheffield, Middlesbrough and Birmingham. Below we list the UTLAs in these areas in descending order.

```r
hotspots <- utla_shp %>% select(ctyu19nm, covid19_r) %>%
  filter(covid19_r > 190)
hotspots[order(-hotspots$covid19_r),]
```

```r
map_utla = tm_shape(utla_shp) +
  tm_fill(col = "covid19_r", title = legend_title, palette = magma(256), style = "cont") + # add fill
  tm_borders(col = "white", lwd = .1) + # add borders
  tm_compass(type = "arrow", position = c("right", "top"), size = 5) + # add compass
  tm_scale_bar(breaks = c(0,1,2), text.size = 0.7, position = c("center", "bottom")) + # add scale
  tm_layout(bg.color = "white") # change background colour
map_utla + tm_shape(reg_shp) + # add region boundaries
  tm_borders(col = "white", lwd = .5) # add borders
```
### 9.5 Global Regression

To provide an intuitive understanding of GWR, a useful start is to explore the data using an ordinary least squares (OLS) linear regression model. The key issue here is to understand if high incidence of COVID-19 is linked to structural differences across UTLAs in England. As indicated above, confirmed positive cases of COVID-19 have been associated with overcrowded housing, vulnerable populations - including people in elderly age groups, economically disadvantaged groups and those suffering from chronic health conditions - ethnic minorities, critical workers in the health & social work, education, accommodation & food, transport, and administrative & support sectors. So, let’s create a set of variables to approximate these factors.

```r
# define predictors
utla_shp <- utla_shp %>%
  mutate(
    crowded_hou = Crwd__ / Hshld,  # share of crowded housing
    elderly = (A_65_ + Ag_85) / Rsdnt,  # share of population aged 65+
    lt_illness = Lng__ / Rsdnt,  # share of population in long-term illness
    ethnic = (Mixed + Indin + Pkstn + Bngld + Chins + Oth_A + Black + Othr_t) / Rsdnt,  # share of nonwhite population
    imd19_ext = IMD20,  # proportion of a larger area's population living in the most deprived LSOAs in the country
    hlthsoc_sec = H___ / E_16__,  # share of workforce in the human health & social work sector
    educ_sec = Edctn / E_16__,  # share of workforce in the education sector
    trnsp_sec = Trn__ / E_16__,  # share of workforce in the Transport & storage sector
    accfood_sec = Ac___ / E_16__,  # share of workforce in the accommodation & food service sector
    admsupport_sec = Adm__ / E_16__,  # share of workforce in the administrative & support sector
    pblic_sec = Pb__ / E_16__  # share of workforce in the public administration & defence sector
  )
```

Let’s quickly examine how they correlate to our outcome variable i.e. incidence rate of COVID-19 using correlation coefficients and correlograms.

```r
# obtain a matrix of Pearson correlation coefficients
df_sel <- st_set_geometry(utla_shp[,37:48], NULL)  # temporary data set removing geometries
cormat <- cor(df_sel, use="complete.obs", method="pearson")
```

### Challenge 1: How does Liverpool ranked in this list?
9.5. GLOBAL REGRESSION

```
sig1 <- corplot::cor.mtest(df_sel, conf.level = .95)

# create a correlogram

# the correlogram shows the strength and significance of the linear relationship
# between our set of variables. The size of the circle reflects the strength
# of the relationships as captured by the Pearson correlation coefficient, and crosses
# indicate statistically insignificant relationships at the 95% level of confidence.
# The colour indicate the direction of the relationship with dark (light) colours
# indicating a negative (positive) association.

The results indicate that the incidence of COVID-19 is significantly and positively related to the share of overcrowded housing, nonwhite ethnic minorities and administrative & support workers. Against expectations, the incidence of COVID-19 appears to be negatively correlated with the share of elderly population, of population suffering from long-term illness and of administrative & support workers, and displays no significant association with the share of the population living in deprived areas as well as the share of public administration.
& defence workers, and health & social workers. The latter probably reflects the effectiveness of the protective measures undertaken to prevent infection among these population groups, but it may also reflect the partial coverage of COVID-19 testing and underreporting. It may also reveal the descriptive limitations of correlation coefficients as they show the relationship between a pairs of variables, not controlling for others. Correlation coefficients can thus produce spurious relationships resulting from confounded variables. We will return to this point below.

The results also reveal high collinearity between particular pairs of variables, notably between the share of crowded housing and of nonwhite ethnic population, the share of crowded housing and of elderly population, the share of overcrowded housing and of administrative & support workers, the share of elderly population and of population suffering from long-term illness. A more refined analysis of multicollinearity is needed. Various diagnostics for multicollinearity in a regression framework exist, including matrix condition numbers (CNs), predictor variance inflation factors (VIFs) and variance decomposition factors (VDPs). Rules of thumb (CNs > 30, VIFs > 10 and VDPs > 0.5) to indicate worrying levels of collinearity can be found in Belsley et al. (2005). To avoid problems of multicollinearity, often a simple strategy is to remove highly correlated predictors. The difficulty is in deciding which predictor(s) to remove, especially when all are considered important. Keep this in mind when specifying your model.

Challenge 2: Analyse the relationship of all the variables executing pairs(df_sel). How accurate would a linear regression be in capturing the relationships for our set of variables?

9.5.1 Global Regression Results

To gain a better understanding of these relationships, we can regress the incidence rate of COVID-19 on a series of factors capturing differences across areas. To focus on the description of GWR, we keep our analysis simple and study the incidence rate of COVID-19 as a function of the share of nonwhite ethnic population and of population suffering from long-term illness by estimating the following OLS linear regression model:

```r
# attach data
attach(utla_shp)

# specify a model equation
eq1 <- covid19_r ~ ethnic + lt_illness
model1 <- lm(formula = eq1, data = utla_shp)

# estimates
summary(model1)
```

##
9.5. GLOBAL REGRESSION

We also compute the VIFs for the variables in the model:

```
vif(model1)
```

The regression results indicate a positive relationship exists between the share of nonwhite population and an increased risk of COVID-19 infection. A one percentage point increase in the share of nonwhite population returns a 271 rise in the cumulative count of COVID-19 infection per 100,000 people, everything else constant. The results also reveal a positive (albeit statistically insignificant) relationship between the share of population suffering from long-term illness and an increased risk of COVID-19 infection, after controlling for the share of nonwhite population, thereby confirming our suspicion about the limitations of correlation coefficients; that is, once differences in the share of nonwhite population are taken into account, the association between the share of population suffering from long-term illness and an increased risk of COVID-19 infection becomes positive. We also test for multicollinearity. The VIFs are below 10 indicating that multicollinearity is not highly problematic.

The \( R^2 \) value for the OLS regression is 0.393 indicating that our model explains only 39% of the variance in the rate of COVID-19 infection. This leaves 71% of the variance unexplained. Some of this unexplained variance can be because we have only included two explanatory variables in our model, but also because the OLS regression model assumes that the relationships in the model are constant over space; that is, it assumes a stationary process. Hence, an OLS regression model is considered to capture global relationships. However, relationships may vary over space. Suppose, for instance, that there are intrinsic behavioural
variations across England and that people have adhered more strictly to self-isolation and social distancing measures in some areas than in others, or that ethnic minorities are less exposed to contracting COVID-19 in certain parts of England. If such variations in associations exist over space, our estimated OLS model will be a misspecification of reality because it assumes these relationships to be constant.

To better understand this potential misspecification, we investigate the model residuals which show high variability (see below). The distribution is non-random displaying large positive residuals in the metropolitan areas of London, Liverpool, Newcastle (in light colours) and the Lake District and large negative residuals across much of England (in black). This conforms to the spatial pattern of confirmed COVID-19 cases with high concentration in a limited number of metropolitan areas (see above). While our residual map reveals that there is a problem with the OLS model, it does not indicate which, if any, of the parameters in the model might exhibit spatial nonstationarity. A simple way of examining if the relationships being modelled in our global OLS model are likely to be stationary over space would be to estimate separate OLS model for each UTLA in England. But this would require higher resolution i.e. data within UTLA, and we only have one data point per UTLA. (2002) (2002, p.40-44) discuss alternative approaches and their limitations.

```r
utla_shp$res_m1 <- residuals(model1)

# map
legend_title = expression("OLS residuals")
map_utla = tm_shape(utla_shp) +
  tm_fill(col = "res_m1", title = legend_title, palette = magma(256), style = "cont") +
  tmBorders(col = "white", lwd = .1) + # add borders
  tm_compass(type = "arrow", position = c("right", "top"), size = 5) + # add compass
  tm_scale_bar(breaks = c(0,1,2), text.size = 0.7, position = c("center", "bottom")) +
  tm_layout(bg.color = "white") # change background colour
map_utla + tm_shape(reg_shp) + # add region boundaries
  tmBorders(col = "white", lwd = .5) # add borders
```
9.6 Fitting a Geographically Weighted Regression

GWR overcomes the limitation of the OLS regression model of generating a global set of estimates. The basic idea behind GWR is to examine the way in which the relationships between a dependent variable and a set of predictors might vary over space. GWR operates by moving a search window from one regression point to the next, working sequentially through all the existing regression points in the dataset. A set of regions is then defined around each regression point and within the search window. A regression model is then fitted to all data contained in each of the identified regions around a regression point, with data points closer to the sample point being weighted more heavily than are those farther away. This process is repeated for all samples points in the dataset. For a data set of 150 observations GWR will fit 150 weighted regression models. The resulting local estimates can then be mapped at the locations of the regression points to view possible variations in the relationships between variables.

Graphically, GWR involves fitting a spatial kernel to the data as described in the Fig. 1. For a given regression point \( X \), the weight \( (W) \) of a data point is at a maximum at the location of the regression point. The weight decreases gradually as the distance between two points increases. A regression model is thus calibrated locally by moving the regression point across the area under study. For each location, the data are weighted differently so that the resulting
estimates are unique to a particular location.

Figure 9.1: Fig. 1. GWR with fixed spatial kernel. Source: Fotheringham et al. (2002, p.45).

9.6.1 Fixed or Adaptive Kernel

A key issue is to decide between two options of spatial kernels: a fixed kernel or an adaptive kernel. Intuitively, a fixed kernel involves using a fixed bandwidth to define a region around all regression points as displayed in Fig. 1. The extent of the kernel is determined by the distance to a given regression point, with the kernel being identical at any point in space. An adaptive kernel involves using varying bandwidth to define a region around regression points as displayed in Fig. 2. The extent of the kernel is determined by the number of nearest neighbours from a given regression point. The kernels have larger bandwidths where the data are sparse.

Figure 9.2: Fig. 2. GWR with adaptive spatial kernel. Source: Fotheringham et al. (2002, p.47).
9.6. FITTING A GEOGRAPHICALLY WEIGHTED REGRESSION

9.6.2 Optimal Bandwidth

A second issue is to define the extent of geographical area (i.e. optimal bandwidth) of the spatial kernel. The bandwidth is the distance beyond which a value of zero is assigned to weight observations. Larger bandwidths include a larger number of observations receiving a non-zero weight and more observations are used to fit a local regression.

To determine the optimal bandwidth, a cross-validation approach is applied; that is, for a location, a local regression is fitted based on a given bandwidth and used to predict the value of the dependent variable. The resulting predicted value is used to compute the residuals of the model. Residuals are compared using a series of bandwidth and the bandwidth returning the smallest local residuals are selected.

**Variance and Bias Trade off**

Choosing an optimal bandwidth involves a compromise between bias and precision. For example, a larger bandwidth will involve using a larger number of observations to fit a local regression, and hence result in reduced variance (or increased precision) but high bias of estimates. On the other hand, too small bandwidth involves using a very small number of observations resulting in increased variance but small bias. An optimal bandwidth offers a compromise between bias and variance.

9.6.3 Shape of Spatial Kernel

Two general set of kernel functions can be distinguished: continuous kernels and kernels with compact support. Continuous kernels are used to weight all observations in the study area and includes uniform, Gaussian and Exponential kernel functions. Kernel with compact support are used to assign a nonzero weight to observations within a certain distance and a zero weight beyond it. The shape of the kernel has been reported to cause small changes to resulting estimates (Brunsdon et al., 1998).

9.6.4 Selecting a Bandwidth

Let’s now implement a GWR model. The first key step is to define the optimal bandwidth. We first illustrate the use of a fixed spatial kernel.

9.6.4.1 Fixed Bandwidth

Cross-validation is used to search for the optimal bandwidth. Recall that this procedure compares the model residuals based on different bandwidths and chooses the optimal solution i.e. the bandwidth returning the smallest model residuals based on a given model specification. A key parameter here is the shape of the geographical weight function (gweight). We set it to be a Gaussian function which is the default. A bi-square function is recommended to
reduce computational time. Since we have a simple model, a Gaussian function
should not take that long. Note that we set the argument `longlat` to `TRUE` and
use latitude and longitude for coordinates (`coords`). When `longlat` is set to
`TRUE`, distances are measured in kilometres.

```r
# find optimal kernel bandwidth using cross validation
fbw <- gwr.sel(eq1,
               data = utla_shp,
               coords = cbind(long, lat),
               longlat = TRUE,
               adapt = FALSE,
               gweight = gwr.Gauss,
               verbose = FALSE)

# view selected bandwidth
fbw
## [1] 29.30417
```

The result indicates that the optimal bandwidth is 39.79 kms. This means that
neighbouring UTLAs within a fixed radius of 39.79 kms will be taken to estimate
local regressions. To estimate a GWR, we execute the code below in which the
optimal bandwidth above is used as an input in the argument `bandwidth`.

```r
# fit a gwr based on fixed bandwidth
fb_gwr <- gwr(eq1,
              data = utla_shp,
              coords = cbind(long, lat),
              longlat = TRUE,
              bandwidth = fbw,
              gweight = gwr.Gauss,
              hatmatrix = TRUE,
              se.fit = TRUE)

fb_gwr
## Call:
## gwr(formula = eq1, data = utla_shp, coords = cbind(long, lat),
##     bandwidth = fbw, gweight = gwr.Gauss, hatmatrix = TRUE, longlat = TRUE,
##     se.fit = TRUE)
## Kernel function: gwr.Gauss
## Fixed bandwidth: 29.30417
## Summary of GWR coefficient estimates at data points:
##          Min.  1st Qu.   Median      3rd Qu.     Max.   Global
## X.Intercept. -187.913  -42.890    93.702     211.685  792.989    63.768
## ethnic        -785.938   104.813   194.609     254.717 1078.854   271.096
## lt_illness     -2599.119   -563.128   128.176     690.603 1507.024   216.198
## Number of data points: 150
```
9.6. FITTING A GEOGRAPHICALLY WEIGHTED REGRESSION

## Effective number of parameters (residual: 2traceS - traceS'S): 57.11019
## Effective degrees of freedom (residual: 2traceS - traceS'S): 92.88981
## Sigma (residual: 2traceS - traceS'S): 38.34777
## Effective number of parameters (model: traceS): 44.65744
## Effective degrees of freedom (model: traceS): 105.3426
## Sigma (model: traceS): 36.00992
## Sigma (ML): 30.17717
## AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): 1580.349
## AIC (GWR p. 96, eq. 4.22): 1492.465
## Residual sum of squares: 136599.2
## Quasi-global R2: 0.7830537

We will skip the interpretation of the results for now and consider them in the next section. Now, we want to focus on the overall model fit and will map the results of the $R^2$ for the estimated local regressions. To do this, we extract the model results stored in a Spatial Data Frame (SDF) and add them to our spatial data frame `utla_shp`. Note that the Quasi-global $R^2$ is very high (0.77) indicating a high in-sample prediction accuracy.

```r
# write gwr output into a data frame
fb_gwr_out <- as.data.frame(fb_gwr$SDF)

utla_shp$fmb_localR2 <- fb_gwr_out$localR2

# map
# Local R2
legend_title = expression("Fixed: Local R2")
map_fbgwr1 = tm_shape(utla_shp) +
  tm_fill(col = "fmb_localR2", title = legend_title, palette = magma(256), style = "cont") + # add fill
  tm_borders(col = "white", lwd = .1) + # add borders
  tm_compass(type = "arrow", position = c("right", "top"), size = 5) + # add compass
  tm_scale_bar(breaks = c(0,1,2), text.size = 0.7, position = c("center", "bottom")) + # add scale bar
  tm_layout(bg.color = "white") # change background colour
map_fbgwr1 + tm_shape(reg_shp) + # add region boundaries
  tm_borders(col = "white", lwd = .5) # add borders
```
The map shows very high in-sample model predictions of up to 80% in relatively large UTLAs (i.e. Cornwall, Devon and Cumbria) but poor predictions in Lincolnshire and small UTLAs in the North West and Yorkshire & The Humber Regions and the Greater London. The spatial distribution of this pattern may reflect a potential problem that arise in the application of GWR with fixed spatial kernels. The use of fixed kernels implies that local regressions for small spatial units may be calibrated on a large number of dissimilar areas, while local regressions for large areas may be calibrated on very few data points, giving rise to estimates with large standard errors. In extreme cases, generating estimates might not be possible due to insufficient variation in small samples. In practice, this issue is relatively common if the number of geographical areas in the dataset is small.

9.6.4.2 Adaptive Bandwidth

To reduce these problems, adaptive spatial kernels can be used. These kernels adapt in size to variations in the density of the data so that the kernels have larger bandwidths where the data are sparse and have smaller bandwidths where the data are plentiful. As above, we first need to search for the optimal bandwidth before estimating a GWR.

```r
# find optimal kernel bandwidth using cross validation
abw <- gwr.sel(eq1,
    data = utla_shp,
    coords=cbind( long, lat),
    longlat = TRUE,
)```
The optimal bandwidth is 0.03 indicating the proportion of observations (or k-nearest neighbours) to be included in the weighting scheme. In this example, the optimal bandwidth indicates that for a given UTLA, 3% of its nearest neighbours should be used to calibrate the relevant local regression; that is about 5 UTLAs. The search window will thus be variable in size depending on the extent of UTLAs. Note that here the optimal bandwidth is defined based on a data point’s k-nearest neighbours. It can also be defined by geographical distance as done above for the fixed spatial kernel. We next fit a GWR based on an adaptive bandwidth.

```r
ab_gwr <- gwr(eq1,
               data = utla_shp,
               coords = cbind(long, lat),
               longlat = TRUE,
               adapt = abw,
               gweight = gwr.Gauss,
               hatmatrix = TRUE,
               se.fit = TRUE)
```

```
   Call: gwr(formula = eq1, data = utla_shp, coords = cbind(long, lat),
             gweight = gwr.Gauss, adapt = abw, hatmatrix = TRUE, longlat = TRUE,
             se.fit = TRUE)
          Kernel function: gwr.Gauss
          Adaptive quantile: 0.03126972 (about 4 of 150 data points)
   Summary of GWR coefficient estimates at data points:
     Min.  1st Qu.  Median    3rd Qu.   Max. Global
ethnic       -121.872  106.822  229.591  283.739  1162.123  271.096
lt_illness   -1907.098 -746.468  -125.855  798.875  1496.549 216.198
   Number of data points: 150
   Effective number of parameters (residual: 2traceS - traceS’S): 48.59361
   Effective degrees of freedom (residual: 2traceS - traceS’S): 101.4064
   Sigma (residual: 2traceS - traceS’S): 36.57493
   Effective number of parameters (model: traceS): 36.04378
```
## Effective degrees of freedom (model: traceS): 113.9562
## Sigma (model: traceS): 34.50222
## Sigma (ML): 30.07257
## AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): 1546.029
## AIC (GWR p. 96, eq. 4.22): 1482.809
## Residual sum of squares: 135653.9
## Quasi-global R2: 0.7845551

### 9.6.5 Model fit

Assessing the global fit of the model, marginal improvements are observed. The $AIC$ and *Residual sum of squares* experienced marginal reductions, while the $R^2$ increased compared to the GRW based on a fixed kernel. To gain a better understanding of these changes, as above, we map the $R^2$ values for the estimated local regressions.

```r
# write gwr output into a data frame
ab_gwr_out <- as.data.frame(ab_gwr$SDF)

utla_shp$amb_ethnic <- ab_gwr_out$ethnic
utla_shp$amb_lt_illness <- ab_gwr_out$lt_illness
utla_shp$amb_localR2 <- ab_gwr_out$localR2

# map
  # Local R2
legend_title = expression("Adaptive: Local R2")
map_abgwr1 = tm_shape(utla_shp) +
  tm_fill(col = "amb_localR2", title = legend_title, palette = magma(256), style = "cont") +
  tm_borders(col = "white", lwd = .1) +
  tm_compass(type = "arrow", position = c("right", "top"), size = 5) +
  tm_scale_bar(breaks = c(0,1,2), text.size = 0.7, position = c("center", "bottom")) +
  tm_layout(bg.color = "white")
map_abgwr1 + tm_shape(reg_shp) +
  tm_borders(col = "white", lwd = .5)"
9.6. FITTING A GEOGRAPHICALLY WEIGHTED REGRESSION

The map reveals notable improvements in local estimates for UTLAs within West and East Midlands, the South East, South West and East of England. Estimates are still poor in hot spot UTLAs concentrating confirmed cases of COVID-19, such as the Greater London, Liverpool and Newcastle areas.

9.6.6 Interpretation

The key strength of GWR models is in identifying patterns of spatial variation in the associations between pairs of variables. The results reveal how these coefficients vary across the 150 UTLAs of England. To examine this variability, let’s first focus on the adaptive GWR output reported in Section 8.6.4.2. The output includes a summary of GWR coefficient estimates at various data points. The last column reports the global estimates which are the same as the coefficients from the OLS regression we fitted at the start of our analysis. For our variable nonwhite ethnic population, the GWR outputs reveals that local coefficients range from a minimum value of -148.41 to a maximum value of 1076.84, indicating that one percentage point increase in the share of nonwhite ethnic population is associated with a reduction of 148.41 in the number of cumulative confirmed cases of COVID-19 per 100,000 people in some UTLAs and an increase of 1076.84 in others. For half of the UTLAs in the dataset, as the share of nonwhite ethnic population increases by one percentage point, the rate of COVID-19 will increase between 106.29 and 291.24 cases; that is, the inter-quartile range between the 1st Qu and the 3rd Qu. To analyse the spatial structure, we next map the estimated coefficients obtained from the adaptive kernel GWR.
# Ethnic

```
legend_title = expression("Ethnic")
map_abgwr2 = tm_shape(utla_shp) +
  tm_fill(col = "amb_ethnic", title = legend_title, palette = magma(256), style = "cont")
  tm_borders(col = "white", lwd = .1) + # add borders
  tm_compass(type = "arrow", position = c("right", "top"), size = 5) + # add compass
  tm_scale_bar(breaks = c(0,1,2), text.size = 0.7, position = c("center", "bottom")) +
  tm_layout(bg.color = "white") # change background colour
map_abgwr2 = map_abgwr2 + tm_shape(reg_shp) + # add region boundaries
  tm_borders(col = "white", lwd = .5) # add borders
```

# Long-term Illness

```
legend_title = expression("Long-term illness")
map_abgwr3 = tm_shape(utla_shp) +
  tm_fill(col = "amb_lt_illness", title = legend_title, palette = magma(256), style = "cont")
  tm_borders(col = "white", lwd = .1) + # add borders
  tm_scale_bar(breaks = c(0,1,2), text.size = 0.7, position = c("center", "bottom")) +
  tm_layout(bg.color = "white") # change background colour
map_abgwr3 = map_abgwr3 + tm_shape(reg_shp) + # add region boundaries
  tm_borders(col = "white", lwd = .5) # add borders
```

`tmap_arrange(map_abgwr2, map_abgwr3)`

Analysing the map for long-term illness, a clear North-South divide can be identified. In the North we observed the expected positive relationship between
COVID-19 and long-term illness i.e. as the share of the local population suffering from long-term illness rises, the cumulative number of positive COVID-19 cases is expected to increase. In the South, we observe the inverse pattern i.e. as the share of local population suffering from long-term illness rises, the cumulative number of positive COVID-19 cases is expected to drop. This pattern is counterintuitive but may be explained by the wider socio-economic disadvantages between the North and the South of England. The North is usually characterised by a persistent concentration of more disadvantaged neighbourhoods than the South where affluent households have tended to cluster for the last 40 years (Rowe et al., 2020).

9.6.7 Assessing statistical significance

While the maps above offer valuable insights to understand the spatial patterning of relationships, they do not identify whether these associations are statistically significant. They may not be. Roughly, if a coefficient estimate has an absolute value of \( t \) greater than 1.96 and the sample is sufficiently large, then it is statistically significant. Our sample has only 150 observations, so we are more conservative and considered a coefficient to be statistically significant if it has an absolute value of \( t \) larger than 2. Note also that p-values could be computed - see Lu et al. (2014).

```r
# compute t statistic
utla_shp$t_ethnic = ab_gwr_out$ethnic / ab_gwr_out$ethnic_se

# categorise t values
utla_shp$t_ethnic_cat <- cut(utla_shp$t_ethnic,
    breaks=c(min(utla_shp$t_ethnic), -2, 2, max(utla_shp$t_ethnic)),
    labels=c("sig", "nonsig", "sig"))

# map statistically significant coefs for ethnic
legend_title = expression("Ethnic: significant")
map_sig = tm_shape(utla_shp) +
    tm_fill(col = "t_ethnic_cat", title = legend_title, legend.hist = TRUE, midpoint = NA, textNA = 1)
    tm_borders(col = "white", lwd = .1) + # add borders
    tm_compass(type = "arrow", position = c("right", "top"), size = 5) + # add compass
    tm_scale_bar(breaks = c(0,1,2), text.size = 0.7, position = c("center", "bottom")) + # add scale
    tm_layout(bg.color = "white", legend.outside = TRUE) # change background colour & place legend

map_sig + tm_shape(reg_shp) + # add region boundaries
    tm_borders(col = "white", lwd = .5) # add borders
```
For the share of nonwhite population, 67% of all local coefficients are statistically significant and these are largely in the South of England. Coefficients in the North tend to be insignificant. Through outliers exist in both regions. In the South, nonsignificant coefficients are observed in the metropolitan areas of London, Birmingham and Nottingham, while significant coefficients exist in the areas of Newcastle and Middlesbrough in the North.

Challenge 3 Compute the t values for the intercept and estimated coefficient for long-term illness and create maps of their statistical significance. How many UTLAs report statistically significant coefficients?

9.6.8 Collinearity in GWR

An important final note is: collinearity tends to be problematic in GWR models. It can be present in the data subsets to estimate local coefficients even when not observed globally Wheeler and Tiefelsdorf (2005). Collinearity can be highly problematic in the case of compositional, categorical and ordinal predictors, and may result in exact local collinearity making the search for an optimal bandwidth
impossible. A recent paper suggests potential ways forward (Comber et al., 2020).

9.7 Questions

We will continue to use the COVID-19 dataset. Please see Chapter 11 for details on the data.

```r
sdf <- st_read("data/assignment_2_covid/covid19_eng.gpkg")
```

Using these data, you are required to address the following challenges:

- Fit a GWR model using a fixed and an adaptive bandwidth.
- Create a multiple map figure to analyse the spatial variation of coefficients.

Analyse and discuss:

1. How regression coefficients vary across space. Do they vary in size and statistical significance?
2. What is the appropriate bandwidth for your GWR? Why?
Chapter 10

Spatio-Temporal Analysis

This chapter provides an introduction to the complexities of spatio-temporal data and modelling. For modelling, we consider the Fixed Rank Kriging (FRK) framework developed by Cressie and Johannesson (2008). It enables constructing a spatial random effects model on a discretised spatial domain. Key advantages of this approach comprise the capacity to: (1) work with large data sets, (2) be scaled up; (3) generate predictions based on sparse linear algebraic techniques, and (4) produce fine-scale resolution uncertainty estimates.

The content of this chapter is based on:

- Wikle et al. (2019), a recently published book which provides a good overview of existing statistical approaches to spatio-temporal modelling and R packages.
- Zammit-Mangion and Cressie (2017), who introduce the statistical framework and R package for modelling spatio-temporal used in this Chapter.

10.1 Dependencies

This chapter uses the following libraries:

```r
# Data manipulation, transformation and visualisation
library(tidyverse)

# Nice tables
library(kableExtra)

# Simple features (a standardised way to encode vector data ie. points, lines, polygons)
library(sf)

# Spatial objects conversion
library(sp)

# Thematic maps
library(tmap)
```
10.2 Data

For this chapter, we will use data on:

- COVID-19 confirmed cases from 30th January, 2020 to 21st April, 2020 from Public Health England via the GOV.UK dashboard;

- resident population characteristics from the 2011 census, available from the Office of National Statistics; and,

- 2019 Index of Multiple Deprivation (IMD) data from GOV.UK and published by the Ministry of Housing, Communities & Local Government. The data are at the ONS Upper Tier Local Authority (UTLA) level - also known as Counties and Unitary Authorities.

For a full list of the variables included in the data sets used in this chapter, see the readme file in the sta data folder.\(^1\) Before we get our hands on the data, there are some important concepts that need to be introduced. They provide a useful framework to understand the complex structure of spatio-temporal data. Let’s start by first highlighting the importance of spatio-temporal analysis.

\(^1\)Read the file in R by executing `read_tsv("data/sta/readme.txt")`
10.3 Why Spatio-Temporal Analysis?

Investigating the spatial patterns of human processes as we have done so far in this book only offers a partial incomplete representation of these processes. It does not allow understanding of the temporal evolution of these processes. Human processes evolve in space and time. Human mobility is an inherent geographical process which changes over the course of the day, with peaks at rush hours and high concentration towards employment, education and retail centres. Exposure to air pollution changes with local climatic conditions, and emission and concentration of atmospheric pollutants which fluctuate over time. The rate of disease spread varies over space and may significantly change over time as we have seen during the current outbreak, with flattened or rapid declining trends in Australia, New Zealand and South Korea but fast proliferation in the United Kingdom and the United States. Only by considering time and space together we can address how geographic entities change over time and why they change. A large part of how and why of such change occurs is due to interactions across space and time, and multiple processes. It is essential to understand the past to inform our understanding of the present and make predictions about the future.

10.3.1 Spatio-temporal Data Structures

A first key element is to understand the structure of spatio-temporal data. Spatio-temporal data incorporate two dimensions. At one end, we have the temporal dimension. In quantitative analysis, time-series data are used to capture geographical processes at regular or irregular intervals; that is, in a continuous (daily) or discrete (only when an event occurs) temporal scale. At another end, we have the spatial dimension. We often use spatial data as temporal aggregations or temporally frozen states (or ‘snapshots’) of a geographical process - this is what we have done so far. Recall that spatial data can be capture in different geographical units, such as areal or lattice, points, flows or trajectories - refer to the introductory lecture in Week 1. Relatively few ways exist to formally integrate temporal and spatial data in consistent analytical framework. Two notable exceptions in R are the packages TraMiner (Gabaldinho et al., 2009) and spacetime (Pebesma et al., 2012). We use the class definitions defined in the R package spacetime. These classes extend those used for spatial data in sp and time-series data in xts. Next a brief introduction to concepts that facilitate thinking about spatio-temporal data structures.

10.3.1.1 Type of Table

Spatio-temporal data can be conceptualised as three main different types of tables:

- time-wide: a table in which columns correspond to different time points
- space-wide: a table in which columns correspond to different spatial location
• long formats: a table in which each row-column pair corresponds to a specific time and spatial location (or space coordinate)

Note that data in long format are space inefficient because spatial coordinates and time attributes are required for each data point. Yet, data in this format are relatively easy to manipulate via packages such as dplyr and tidyr, and visualise using ggplot2. These packages are designed to work with data in long format.

10.3.1.2 Type of Spatio-Temporal Object

To integrate spatio-temporal data, spatio-temporal objects are needed. We consider four different spatio-temporal frames (STFs) or objects which can be defined via the package spacetime:

• Full grid (STF): an object containing data on all possible locations in all time points in a sequence of data;

• Sparse grid (STS): an object similar to STF but only containing non-missing space-time data combinations;

• Irregular (STI): an object with an irregular space-time data structure, where each point is allocated a spatial coordinate and a time stamp;

• Simple Trajectories (STT): an object containing a sequence of space-time points that form trajectories.

More details on these spatio-temporal structures, construction and manipulation, see Pebesma et al. (2012). Enough theory, let’s code!

10.4 Data Wrangling

This section illustrates the complexities of handling spatio-temporal data. It discusses good practices in data manipulation and construction of a Space Time Irregular Data Frame (STIDF) object. Three key requirements to define a STFDF object are:

1. Have a data frame in long format i.e. a location-time pair data frame

2. Define a time stamp

3. Construct the spatio-temporal object of class STIDF by indicating the spatial and temporal coordinates

Let’s now read all the required data. While we can have all data in a single data frame, you will find helpful to have separate data objects to identify:

• spatial locations

• temporal units

• data
These data objects correspond to `locs`, `time`, and `covid19` and `censusimd` below. Throughout the chapter you will notice that we switch between the various data frames when convenient, depending on the operation.

```r
# clear workspace
rm(list=ls())

# read ONS UTLA shapefile
utla_shp <- st_read("data/sta/ons_utla.shp")
```

```r
# create table of locations
locs <- utla_shp %>%
  as.data.frame() %>%
  dplyr::select(object, cty19c, ctyu19nm, long, lat, st_rs)

# read time data frame
time <- read_csv("data/sta/reporting_dates.csv")

# read COVID-19 data in long format
covid19 <- read_csv("data/sta/covid19_cases.csv")

# read census and IMD data
censusimd <- read_csv("data/sta/2011census_2019imd_utla.csv")
```

If we explore the structure of the data via `head` and `str`, we can see we have data on daily and cumulative new COVID-19 cases for 150 spatial units (i.e. UTLAs) over 71 time points from January 30th to April 21st. We also have census and IMD data for a range of attributes.

```r
head(covid19, 3)
```

```
# A tibble: 3 x 6
##       Area.name Area.code Area.type date       Daily.lab.confirmed Cumulative.lab.confirmed
##   <chr>        <chr>       <chr>       <date>               <dbl>                           <dbl>
## 1 Barking     E09000002 Upper tier 2020-01-30 0                 0
## 2 Barnet      E09000003 Upper tier 2020-01-30 0                 0
## 3 Barnsley    E08000016 Upper tier 2020-01-30 0                 0
```

Once we have understood the structure of the data, we first need to confirm if the `covid19` data are in wide or long format. Luckily they are in long format; otherwise, we would have needed to transform the data from wide to long format. Useful functions to achieve this include `pivot_longer` (pivot_longer) which
has superseded `gather` (spread) in the `tidyrr` package. Note that the `covid19` data frame has 10,650 observations (i.e., rows); that is, 150 UTLAs * 71 daily observations.

We then define a regular time stamp for our temporal data. We use the `lubridate` package to do this. A key advantage of `lubridate` is that it automatically recognises the common separators used when recording dates ("-", "/", ",", and "). As a result, you only need to focus on specifying the order of the date elements to determine the parsing function applied. Below we check the structure of our time data, define a time stamp and create separate variables for days, weeks, months and year.

Note that working with dates can be a complex task. A good discussion of these complexities is provided here.

```r
# check the time structure used for reporting covid cases
head(covid19$date, 5)
```

```r
## [1] "2020-01-30" "2020-01-30" "2020-01-30" "2020-01-30" "2020-01-30"
```

```r
# parsing data into a time stamp
covid19$date <- ymd(covid19$date)
class(covid19$date)
## [1] "Date"
```

```r
# separate date variable into day, week, month and year variables
covid19$day <- day(covid19$date)
covid19$week <- week(covid19$date)  # week of the year
covid19$month <- month(covid19$date)
covid19$year <- year(covid19$date)
```

Once defined the time stamp, we need to add the spatial information contained in our shapefile to create a spatio-temporal data frame.

```r
# join dfs
covid19_spt <- left_join(utla_shp, covid19, by = c("ctyu19nm" = "Area.name"))
```

We now have all the components to build a spatio-temporal object of class `STIDF` using `STIDF` from the `spacetime` package:

```r
# identifying spatial fields
spat_part <- as(dplyr::select(covid19_spt, -c(bng_e, bng_n, Area.code, Area.type, Daily.lab.confirmed.cases, Cumulative.lab.confirmed.cases, date, day, week, month, year)), Class = "Spatial")
```

```r
## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj = prefer_proj): Discarded datum Unknown based on Airy 1830 ellipsoid in Proj4 definition
```

```r
## Warning in showSRID(SRS_string, format = "PROJ", multiline = "NO", prefer_proj = prefer_proj): Discarded datum D_unknown in Proj4 definition
```
# identifying temporal fields
temp_part <- covid19_spt$date

# identifying data
covid19_data <- covid19_spt %>% dplyr::select(c(Area.code, Area.type, date, Daily.lab.confirmed.cases, Cumulative.lab.confirmed.cases, day, week, month, year)) %>% as.data.frame()

# construct STIDF object
covid19_stobj <- STIDF(sp = spat_part, time = temp_part, data = covid19_data)

class(covid19_stobj)

## [1] "STIDF"
## attr(,"package")
## [1] " spacetime"

We now add census and IMD variables. For the purposes of this Chapter, we only add total population and long-term sick or disabled population counts. You can add more variables by adding their names in the select function.

# select pop data
pop <- censusimd %>% dplyr::select("UTLA19NM", "Residents", "Longterm_sick_or_disabled")

# join dfs
covid19_spt <- left_join(covid19_spt, pop, by = c("ctyu19nm" = "UTLA19NM"))
covid19 <- left_join(covid19, pop, by = c("Area.name" = "UTLA19NM"))

## 10.5 Exploring Spatio-Temporal Data

We now have all the required data in place. In this section various methods of data visualisation are illustrated before key dimensions of the data are explored. Both of these types of exploration can be challenging as one or more dimensions in space and one in time need to be interrogated.

### 10.5.1 Visualisation

In the context spatio-temporal data, a first challenge is data visualization. Visualising more than two dimensions of spatio-temporal data, so it is helpful to slice or aggregate the data over a dimension, use color, or build animations through time. Before exploring the data, we need to define our key variable of interest; that is, the number of confirmed COVID-19 cases per 100,000 people. We also compute the cumulative number of confirmed COVID-19 cases per 100,000 people as it may be handy in some analyses.
First create variable to be analysed:

```r
# rate of new covid-19 infection
covid19_spt$n_covid19_r <- round( (covid19_spt$Daily.lab.confirmed.cases / covid19_spt$Residents) * 100000)

covid19$n_covid19_r <- round( (covid19$Daily.lab.confirmed.cases / covid19$Residents) * 100000)
```

```r
# risk of cumulative covid-19 infection
covid19_spt$c_covid19_r <- round( (covid19_spt$Cumulative.lab.confirmed.cases / covid19_spt$Residents) * 100000)

covid19$c_covid19_r <- round( (covid19$Cumulative.lab.confirmed.cases / covid19$Residents) * 100000)
```

### 10.5.1.1 Spatial Plots

One way to visualise the data is using spatial plots; that is, snapshots of a geographic process for a given time period. Data can be mapped in different ways using clorepleth, countour or surface plots. The key aim of these maps is to understand how the overall extent of spatial variation and local patterns of spatial concentration change over time. Below we visualise the weekly number of confirmed COVID-19 cases per 100,000 people.

Note that Weeks range from 5 to 16 as they refer to calendar weeks. Calendar week 5 is when the first COVID-19 case in England was reported.

```r
# create data frame for new cases by week
daycases_week <- covid19_spt %>%
  group_by(week, ctyu19nm, as.character(cty19c), Residents) %>%
  summarise(n_daycases = sum(Daily.lab.confirmed.cases))
```

`summarise()` has grouped output by 'week', 'ctyu19nm', 'as.character(cty19c)'. You can override using the `.groups` argument.

```r
# weekly rate of new covid-19 infection
daycases_week$wn_covid19_r <- (daycases_week$n_daycases / daycases_week$Residents) * 100000
```

```r
# map
legend_title = expression("Cumulative Cases per 100,000 Population")
tm_shape(daycases_week) +
  tm_fill("wn_covid19_r", title = legend_title, palette = magma(256), style = "cont", legend = TRUE) +
  tm_facets(by = "week", ncol = 4) +
  tm_borders(col = "white", lwd = .1) + # add borders +
  tm_layout(bg.color = "white", # change background colour
             legend.outside = TRUE, # legend outside
             legend.outside.position = "bottom",
             legend.stack = "horizontal",
             legend.title.size = 2,
             legend.width = 1,
             legend.height = 1,
             panel.label.size = 3,
             main.title = "New COVID-19 Cases by Calendar Week, UTLA, England")
```
10.5. EXPLORING SPATIO-TEMPORAL DATA

## Warning in pre_process_gt(x, interactive = interactive, orig_crs =
## gm$shape.orig_crs): legend.width controls the width of the legend within a map.
## Please use legend.outside.size to control the width of the outside legend

New COVID-19 Cases by Calendar Week, UTLA, England

<table>
<thead>
<tr>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Map" /></td>
<td><img src="image2.png" alt="Map" /></td>
<td><img src="image3.png" alt="Map" /></td>
<td><img src="image4.png" alt="Map" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5.png" alt="Map" /></td>
<td><img src="image6.png" alt="Map" /></td>
<td><img src="image7.png" alt="Map" /></td>
<td><img src="image8.png" alt="Map" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image9.png" alt="Map" /></td>
<td><img src="image10.png" alt="Map" /></td>
<td><img src="image11.png" alt="Map" /></td>
<td><img src="image12.png" alt="Map" /></td>
</tr>
</tbody>
</table>

Cumulative Cases per 100,000 Population

The series of maps reveal a stable pattern of low reported cases from calendar weeks 5 to 11. From week 12 a number of hot spots emerged, notably in London, Birmingham, Cumbria and subsequently around Liverpool. The intensity of new cases seem to have started to decline from week 15; yet, it is important to note that week 16 display reported cases for only two days.

10.5.1.2 Time-Series Plots

Time-series plots can be used to capture a different dimension of the process in analysis. They can be used to better understand changes in an observation location, an aggregation of observations, or multiple locations simultaneously over time. We plot the cumulative number of COVID-19 cases per 100,000 people for UTLAs reporting over 310 cases. The plots identify the UTLAs in
London, Newcastle and Sheffield reporting the largest numbers of COVID-19 cases. The plots also reveal that there has been a steady increase in the number of cases, with some differences. While cases have steadily increase in Brent and Southwark since mid March, the rise has been more sudden in Sunderland. The plots also reveal a possible case of misreporting in Sutton towards the end of the series.

```r
tsp <- ggplot(data = covid19_spt, 
              mapping = aes(x = date, y = c_covid19_r, 
                             group = ctyu19nm))
tsp + geom_line(color = "blue") + 
gghighlight(max(c_covid19_r) > 310, use_direct_label = FALSE) + 
labs(title= paste(" "), x="Date", y="Cumulative Cases per 100,000") + 
theme_classic() + 
theme(plot.title=element_text(size = 20)) + 
theme(axis.text=element_text(size=16)) + 
theme(axis.title.y = element_text(size = 18)) + 
theme(axis.title.x = element_text(size = 18)) + 
theme(plot.subtitle=element_text(size = 16)) + 
theme(axis.title=element_text(size=20, face="plain")) + 
facet_wrap(~ ctyu19nm)
```
10.5. EXPLORING SPATIO-TEMPORAL DATA

10.5.1.3 Hovmöller Plots

An alternative visualisation is a Hovmöller plot - sometimes known as heatmap. It is a two-dimensional space-time representation in which space is collapsed onto one dimension against time. Hovmöller plots can easily be generated if the data are arranged on a space-time grid; however, this is rarely the case. Luckily we have **ggplot**! which can do magic rearranging the data as needed. Below we produce a Hovmöller plot for UTLAs with resident populations over 260,000. The plot makes clear that the critical period of COVID-19 spread has been during April despite the implementation of a series of social distancing measures by the government.

```r
ggplot(data = dplyr::filter(covid19_spt, Residents > 260000),
      mapping = aes(x = date, y = reorder(ctyu19nm, c_covid19_r), fill = c_covid19_r)) +
  geom_tile() +
  scale_fill_viridis(name = "New Cases per 100,000", option = "plasma", begin = 0, end = 1, direction = 1)
```

...
theme_minimal() +
labs(title = paste(" "), x="Date", y="Upper Tier Authority Area") +
theme(legend.position = "bottom") +
theme(legend.title = element_text(size=15)) +
theme(axis.text.y = element_text(size=10)) +
theme(axis.text.x = element_text(size=15)) +
theme(axis.title=element_text(size=20, face="plain")) +
theme(legend.key.width = unit(5, "cm"), legend.key.height = unit(2, "cm"))

10.5.1.4 Interactive Plots

Interactive visualisations comprise very effective ways to understand spatio-temporal data and they are now fairly accessible. Interactive visualisations allow for a more data-immersive experience, and enable exploration of the data without having to resort to scripting. Here is when the use of tmap shines as it does not only enables easily creating nice static maps but also interactive maps!
10.5. EXPLORING SPATIO-TEMPORAL DATA

Below an interactive map for a time snapshot of the data (i.e. 2020-04-14) is produced, but with a bit of work layers can be added to display multiple temporal slices of the data.

```r
# map
legend_title = expression("Cumulative Cases per 100,000 Population")
imap = tm_shape(dplyr::filter(covid19_spt[,c("ctyu19nm", "date", "c_covid19_r")], as.character(date) == "2020-04-14"),
  as.character(date) == "2020-04-14"),
  labels = "Area.name"
+ tm_fill("c_covid19_r", title = legend_title, palette = magma(256), style = "cont", legend.is.portrait = FALSE,
  alpha = 0.7)
+ tm_borders(col = "white")
+ #tm_text("ctyu19nm", size = .4) +
  tm_layout(bg.color = "white", # change background colour
  legend.outside = TRUE, # legend outside
  legend.title.size = 1,
  legend.width = 1)
```

To view the map on your local machines, execute the code chunk below removing the # sign.

```r
#tm_map_mode("view")
#imap
```

Alternative data visualisation tools are animations, telliscope and shiny. Animations can be constructed by plotting spatial data frame-by-frame, and then stringing them together in sequence. A useful R packages `gganimate` and `tmap`! See Lovelace et al. (2020). Note that the creation of animations may require external dependencies; hence, they have been included here. Both `telliscope` and `shiny` are useful ways for visualising large spatio-temporal data sets in an interactive ways. Some effort is required to deploy these tools.

10.5.2 Exploratory Analysis

In addition to visualising data, we often want to obtain numerical summaries of the data. Again, innovative ways to reduce the inherent dimensionality of the data and examine dependence structures and potential relationships in time and space are needed. We consider visualisations of empirical spatial and temporal means, dependence structures and some basic time-series analysis.

10.5.2.1 Means

Empirical Spatial Mean

The empirical spatial mean for a data set can be obtained by averaging over time points for one location. In our case, we can compute the empirical spatial mean by averaging the daily rate of new COVID-19 cases for UTLAs between January 30th and April 21st. It reveals that Brent, Southwark and Sunderland report an average daily infection rate of over 5 new cases per 100,000 people, whereas Rutland and Isle of Wight display an average of less than 1.
CHAPTER 10. SPATIO-TEMPORAL ANALYSIS

```r
# compute empirical spatial mean
sp_av <- covid19_spt %>% group_by(ctyu19nm) %>% summarise(sp_mu_emp = mean(n_covid19_r))

# plot empirical spatial mean
ggplot(data=sp_av) + geom_col(aes(y = reorder(ctyu19nm, sp_mu_emp), x = sp_mu_emp, fill = "grey50")) + theme_classic() + labs(title = paste(" "), x = "Average New Cases per 100,000", y = "Upper Tier Authority Area") + theme(legend.position = "bottom") + theme(axis.text.y = element_text(size = 7)) + theme(axis.text.x = element_text(size = 12)) + theme(axis.title = element_text(size = 20, face = "plain"))
```

Empirical Temporal Mean

![Graph showing empirical temporal mean)](image_url)
The empirical temporal mean for a data set can be obtained by averaging across spatial locations for a time point. In our case, we can compute the empirical temporal mean by averaging the rate of new COVID-19 cases over UTLAs by day. The empirical temporal mean is plotted below revealing a peak of 8.32 number of new cases per 100,000 people the 7th of April, steadily decreasing to 0.35 for the last reporting observation in our data; that is, April 21st.

```r
# compute temporal mean
tm_av <- covid19 %>%
  group_by(date) %>%
  summarise(tm_mu_emp = mean(n_covid19_r))

# plot temporal mean + trends for all spatial units
ggplot() +
  geom_line(data = covid19, mapping = aes(x = date, y = n_covid19_r,
                                         group = Area.name), color = "gray80") +
  theme_classic() +
  geom_smooth(data = tm_av, mapping = aes(x = date, y = tm_mu_emp),
              alpha = 0.5,
              se = FALSE) +
  labs(title= paste(" "), x="Date", y="Cumulative Cases per 100,000") +
  theme_classic() +
  theme(plot.title=element_text(size = 18)) +
  theme(axis.text=element_text(size=14)) +
  theme(axis.title.y = element_text(size = 16)) +
  theme(axis.title.x = element_text(size = 16)) +
  theme(plot.subtitle=element_text(size = 16)) +
  theme(axis.title=element_text(size=18, face="plain"))
```

Note the empirical temporal mean is smoothed via local polynomial regression fitting; hence below zero values are reported between February and March.
10.5.2.2 Dependence

Spatial Dependence

As we know spatial dependence refers to the spatial relationship of a variable’s values for a pairs of locations at a certain distance apart, so that are more similar (or less similar) than expected for randomly associated pairs of observations. Patterns of spatial dependence may change over time. In the case of a disease outbreak patterns of spatial dependence can change very quickly as new cases emerge and social distancing measures are implemented. Chapter 6 illustrates how to measure spatial dependence in the context of spatial data.

Challenge 1: Measure how spatial dependence change over time. Hint: compute the Moran’s I on the rate of new COVID-19 cases (i.e. `n_covid19_r` in the `covid19` data frame) at multiple time points.
Note: recall that the problem of ignoring the dependence in the errors when doing OLS regression is that the resulting standard errors and prediction standard errors are inappropriate. In the case of positive dependence, which is the most common case in spatio-temporal data (recall Tobler’s law), the standard errors and prediction standard errors are underestimated. This is if dependence is ignored, resulting in a false sense of how good the estimates and predictions really are.

**Temporal Dependence**

As for spatial data, dependence can also exists in temporal data. Temporal dependence or temporal autocorrelation exists when a variable’s value at time $t$ is dependent on its value(s) at $t-1$. More recent observations are often expected to have a greater influence on present observations. A key difference between temporal and spatial dependence is that temporal dependence is unidirectional (i.e. past observations can only affect present or future observations but not inversely), while spatial dependence is multidirectional (i.e. an observation in a spatial unit can influence and be influenced by observations in multiple spatial units).

Before measuring the temporal dependence is our time-series, a time-series object needs to be created with a time stamp and given cycle frequency. A cycle frequency refers to when a seasonal pattern is repeated. We consider a time series of the total number of new COVID-19 cases per 100,000 (i.e. we sum cases over UTLAs by day) and the frequency set to 7 to reflect weekly cycles. So we end up with a data frame of length 71.

```r
# create a time series object
total_cnt <- covid19 %>% group_by(date) %>%
  summarise(new_cases = sum(n_covid19_r))
total_cases_ts <- ts(total_cnt$new_cases,
  start = 1,
  frequency = 7)
```

There are various ways to test for temporal autocorrelation. An easy way is to compute the correlation coefficient between a time series measured at time $t$ and its lag measured at time $t-1$. Below we measure the temporal autocorrelation in the rate of new COVID-19 cases per 100,000 people. A correlation of 0.97 is returned indicating high positive autocorrelation; that is, high (low) past numbers of new COVID-19 cases per 100,000 people tend to correlate with higher (lower) present numbers of new COVID-19 cases. The Durbin-Watson test is often used to test for autocorrelation in regression models.

```r
# create lag term t-1
lag_new_cases <- total_cnt$new_cases[-1]
total_cnt <- cbind(total_cnt[1:70,], lag_new_cases)
cor(total_cnt[,2:3])
```
## new_cases lag_new_cases
## new_cases 1.000000 0.974284
## lag_new_cases 0.974284 1.000000

### Time Series Components

In addition to temporal autocorrelation, critical to the analysis of time-series are its constituent components. A time-series is generally defined by three key components:

- **Trend**: A trend exists when there is a long-term increase or decrease in the data.

- **Seasonal**: A seasonal pattern exists when a time series is affected by seasonal factors and is of a fixed and known frequency. Seasonal cycles can occur at various time intervals such as the time of the day or the time of the year.

- **Cyclic (random)**: A cycle exists when the data exhibit rises and falls that are not of a fixed frequency.

To understand and model a time series, these components need to be identified and appropriately incorporated into a regression model. We illustrate these components by decomposing our time series for total COVID-19 cases below. The top plot shows the observed data. Subsequent plots display the trend, seasonal and random components of the total number of COVID-19 cases on a weekly periodicity. They reveal a clear inverted U-shape trend and seasonal pattern. This idea that we can decompose data to extract information and understand temporal processes is key to understand the concept of basis functions to model spatio-temporal data, which will be introduced in the next section.

```r
# decompose time series
dec_ts <- decompose(total_cases_ts)
# plot time series components
plot(dec_ts)
```
10.6 Spatio-Temporal Data Modelling

Having some understanding of the spatio-temporal patterns of COVID-19 spread through data exploration, we are ready to start further examining structural relationships between the rate of new infections and local contextual factors via regression modelling across UTLAs. Specifically we consider the number of new cases per 100,000 people to capture the rate of new infections and only one contextual factor; that is, the share of population suffering from long-term sickness or disabled. We will consider some basic statistical models, of the form of linear regression and generalized linear models, to account for spatio-temporal dependencies in the data. Note that we do not consider more complex structures based on hierarchical models or spatio-temporal weighted regression models which would be the natural step moving forward.

As any modelling approach, spatio-temporal statistical modelling has three principal goals:

1. predicting values of a given outcome variable at some location in space within the time span of the observations and offering information about the uncertainty of those predictions;

2. performing statistical inference about the influence of predictors on an outcome variable in the presence of spatio-temporal dependence; and,
3. forecasting future values of an outcome variable at some location, offering information about the uncertainty of the forecast.

10.6.1 Intuition

The key idea on what follows is to use a basic statistical regression model to understand the relationship between the share of new COVID-19 infections and the share of population suffering from long-term illness, accounting for spatio-temporal dependencies. We will consider what is known as a trend-surface regression model which assumes that spatio-temporal dependencies can be accounted for by “trend” components and incorporate as predictors in the model. Formally we consider the regression model below which seeks to account for spatial and temporal trends.

\[ y(s_i, t_j) = \beta_0 + \beta_k x(s_i, t_j) + e(s_i, t_j) \]

where \( \beta_0 \) is the intercept and \( \beta_k \) represents a set of regression coefficients associated with \( x(s_i, t_j) \); the \( k \) indicates the number of covariates at spatial location \( s_i \) and time \( t_j \); \( e \) represents the regression errors which are assumed to follow a normal distribution. The key difference to approaches considered in previous chapters is the incorporation of space and time together. As we learnt from the previous section, this has implications are we now have two sources of dependence: spatial and temporal autocorrelation, as well as seasonal and trend components. This has implications for modelling as we now need to account for all of these components if we are to establish any relationship between \( y \) and \( x \).

A key implication is how we consider the set of covariates represented by \( x \). Three key types can be identified:

- spatial-variant, temporal-invariant covariates: these are attributes which may vary across space but be temporally invariant, such as geographical distances;
- spatial-invariant, temporal-variant covariates: these are attributes which do not vary across space but change over time; and,
- spatial-variant, temporal-variant covariates: these are attributes which vary over both space and time;

Note that what is variant or invariant will depend on the spatial and temporal scale of the analysis.

We can also consider spatio-temporal “basis functions”. Note that this is an important concept for the rest of the Chapter. What are basis functions then? If you think that spatio-temporal data represent a complex set of curves or surfaces in space, basis functions represent the components into which this set of curves can be decomposed. In this sense, basis functions operate in a similar fashion as the decomposition of time series considered above i.e. time series data
can be decomposed into a trend, seasonal and random components and their sum can be used to represent the observed temporal trajectory. Basis functions offer an effective way to incorporate spatio-temporal dependencies. Thus, basis functions have the key goal of accounting for spatio-temporal dependencies as spatial weight matrices or temporal lags help accounting spatial autocorrelation in spatial models and temporal autocorrelation in time series analysis.

As standard regression coefficients, basis functions are related to \( y \) via coefficients (or weights). The difference is that we typically assume that basis functions are known while coefficients are random. Examples of basis functions include polynomials, splines, wavelets, sines and cosines so various linear combinations that can be used to infer potential spatio-temporal dependencies in the data. This is similar to deep learning models in which cases you provide, for example, an image and the model provides a classification of pixels. But you normally do not know what the classification represents (hence they are known as black boxes!) so further analysis on the classification is needed to understand what the model has attempted to capture. Basically basis functions are smoother functions to represent the observed data, and their objective to capture the spatial and temporal variability in the data as well as their dependence.

For our application, we start by considering a basic OLS regression model with the following basis functions to account spatial-temporal structures:

- overall mean;
- linear in lon-coordinate;
- linear in lat-coordinate;
- linear time daily trend;
- additional spatio-temporal basis functions which are presented below; and,

These basis functions are incorporated as independent variables in the regression model. Additionally, we also include the share of population suffering from long-term illness as we know it is highly correlated to the cumulative number of COVID-19 cases. The share of population suffering long-term illness is incorporated as a spatial-variant, temporal-invariant covariates given that rely in 2011 census data.

10.6.2 Fitting Spatio-Temporal Models

As indicated at the start of this Chapter, we use the FRK framework developed by Cressie and Johannesson (2008). It provides a scalable, relies on the use a spatial random effects model (with which we have some familiarity) and can be easily implemented in R by the use of the FRK package (Zammit-Mangion and Cressie, 2017). In this framework, a spatially correlated errors can be decomposed using a linear combination of spatial basis functions, effectively addressing issues of spatial-temporal dependence and nonstationarity. The specification of spatio-temporal basis functions is a key component of the model and they can be generated automatically or by the user via the FRK package. We will use the
automatically generated functions. While as we will notice they are difficult to interpret, user generated functions require greater understanding of the spatio-temporal structure of COVID-19 which is beyond the scope of this Chapter.

Prepare Data

The first step to create a data frame with the variables that we will consider for the analysis. We first remove the geometries to convert `covid19_spt` from a simple feature object to a data frame and then compute the share of long-term illness population.

```r
# remove geometries
st_geometry(covid19_spt) <- NULL

# share of population in long-term illness
covid19_spt <- covid19_spt %>%
mutate(
  lt_illness = Longterm_sick_or_disabled / Residents
)
```

Construct Basis Functions

We now build the set of basis functions. The can be constructed by using the function `auto_basis` from the FRK package. The function takes as arguments: data, nres (which is the number of “resolutions” or aggregation to construct); and type of basis function to use. We consider a single resolution of the default Gaussian radial basis function.

```r
# build basis functions
G <- auto_basis(data = covid19_spt[,c("long","lat")]) %>%
  SpatialPoints(), # To sp obj
  nres = 1, # One resolution
  type = "Gaussian") # Gaussian BFs

S <- eval_basis(basis = G, # basis functions
  s = covid19_spt[,c("long","lat")]) %>%
  as.matrix() %>%
  as.matrix() # conv. to matrix

colnames(S) <- paste0("B", 1:ncol(S)) # assign column names
```

Add Basis Functions to Data Frame

We then prepare a data frame for the regression model, adding the weights extracted from the basis functions. These weights enter as covariates in our model. Note that the resulting number of basis functions is nine. Explore by executing `colnames(S)`. Below we select only relevant variables for our model.

```r
# selecting variables
reg_df <- cbind(covid19_spt, S) %>%
  dplyr::select(ctyu19nm, c_covid19_r, long, lat, day, lt_illness, B1:B9)
```
10.6. SPATIO-TEMPORAL DATA MODELLING

Fit Linear Regression

We now fit a linear model using \texttt{lm} including as covariates longitude, latitude, day, share of long-term ill population and the nine basis functions.

Recall that latitude refers to north/south from the equator and longitude refers to west/east from Greenwich. Further up north means a higher latitude score. Further west means higher longitude score. Scores for Liverpool (53.4084° N, 2.9916° W) are thus higher than for London (51.5074° N, 0.1278° W). This will be helpful for interpretation.

\[
\texttt{eq1 <- c\_covid19\_r \sim long + lat + day + lt\_illness + .}
\]
\[
\texttt{lm\_m <- lm(formula = eq1,}
\texttt{ data = dplyr::select(reg\_df, -ctyu19nm))}
\]
\[
\texttt{lm\_m %>% summary()}
\]

\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
\#
Coefficients for explicitly specified spatial and temporal variables and the share
of long-term ill population are all statistically significant. The interpretation of
the regression coefficients is as usual; that is, one unit increase in a covariate
relates to one unit increase in the dependent variable. For instance, the coeffi-
cient for long-term illness population indicates that UTLAs with a larger share
of long-term ill population in one percentage point tend to have 709 more new
COVID-19 cases per 100,000 people! on average. The coefficient for day reveals
a strong negative temporal dependence with smaller number of new cases per
100,000 people as we move over time. The coefficient for latitude indicates as
we move north the number of new COVID-19 cases per 100,000 people tends to
be higher but lower if we move west.

While overall the model provides some understanding of the spatio-temporal
structure of the spread of COVID-19, the overall fit of the model is relatively
poor. The $R^2$ reveals that the model explains only 5% of the variability of
the spread of COVID-19 cases. Also, except for one, the coefficients associated
to the basis functions are statistically insignificant. A key issue that we have
ignored so far is the fact that our dependent variable is a count and is highly
skewed - refer back to Section [8.4 Exploratory Analysis].

Challenge 2: Explore a model with only spatial components
(i.e. long and lat) or only temporal components (day). What
model returns the largest $R^2$?

Poisson Regression

A Poisson regression model provides a more appropriate framework to address
these issues. We do this fitting a general linear model (or GLM) specifying the
family function to be a Poisson.

```r
# estimate a poisson model
poisson_m1 <- glm(eq1,
      family = poisson("log"), data = dplyr::select(reg_df, -ctyu19nm))
poisson_m1 %>% summary()
```

## Call:
```r
glm(formula = eq1, family = poisson("log"), data = dplyr::select(reg_df, -ctyu19nm))
```

## Deviance Residuals:
```r
  Min      1Q  Median      3Q     Max
```

## Coefficients:
```r
                Estimate Std. Error   z value  Pr(>|z|)  
(Intercept) -9.726e+01    1.010e+00 -96.3137 < 2e-16 ***
```
## Spatial-Temporal Data Modelling

The model seems to provide a better fit to the data as the median of deviance residuals (-6.3) is smaller than for the linear regression model (-27.51). And, all coefficients are positive and statistically significant. Yet, the Poisson model assumes that the mean and variance of the COVID-19 cases is the same. But, given the distribution of our dependent variable, its variance is likely to be greater than the mean. That means the data exhibit “overdispersion”. How do we know this? An estimate of the dispersion is given by the ratio of the deviance to the total degrees of freedom (the number of data points minus the number of covariates). In this case the dispersion estimate is:

\[
\frac{\text{deviance}}{\text{df.residual}}
\]

which is clearly greater than 1! i.e. the data are overdispersed.

### Quasipoisson Regression

An approach to account for overdispersion is to use quasipoisson when calling `glm`. The quasi-Poisson model assumes that the variance is proportional to the mean, and that the constant of the proportionality is the over-dispersion parameter.

```r
# estimate a quasipoisson model
qpoisson_m1 <- glm(eq1, data = df, family = quasipoisson)
```
family = quasipoisson("log"), # QuasiPoisson + log link
data = dplyr::select(reg_df, -ctyu19nm))
quoisson_m1 %>% summary()

##
## Call:
## glm(formula = eq1, family = quasipoisson("log"), data = dplyr::select(reg_df, #
## -ctyu19nm))
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## (Intercept) -97.261261 10.069209 -9.659 < 2e-16 ***
## long -0.978924 0.223534 -4.379 1.20e-05 ***
## lat 1.766675 0.189682 9.314 < 2e-16 ***
## day -0.014405 0.001659 -8.683 < 2e-16 ***
## lt_illness 14.420396 1.845247 7.815 6.02e-15 ***
## B1 6.197505 2.011637 3.081 0.00207 **
## B2 0.361556 0.786651 0.460 0.64580
## B3 11.363202 1.413688 8.038 1.01e-15 ***
## B4 -1.975530 1.445504 -1.367 0.17176
## B5 2.813267 1.251510 2.248 0.02460 *
## B6 -1.333746 1.376430 -0.969 0.33257
## B7 0.769599 2.110944 0.365 0.71544
## B8 -0.709375 1.541475 -0.460 0.64539
## B9 1.809393 1.611435 1.123 0.26153
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 99.42153)
##
## Null deviance: 1032743 on 10649 degrees of freedom
## Residual deviance: 959940 on 10636 degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6

Negative Binomial Regression

The model output indicates major improvement in terms of model fit as the residual deviance (959940) and median of deviance residuals (-6.380) remain unchanged. An alternative approach is a Negative Binomial Model (NBM). This model relaxes the assumption of equality between the mean and variance.
We estimate a NBM by using the function `glm.nb` from the `MASS` package.

```r
# estimate a negative binomial model
nb_m1 <- glm.nb(eq1,
                 data = dplyr::select(reg_df, -ctyu19nm))

nb_m1
```

```r
##
## Call: glm.nb(formula = eq1, data = dplyr::select(reg_df, -ctyu19nm),
## init.theta = 11051670.9, link = log)
##
## Coefficients:
##            (Intercept)      long      lat      day lt_illness       B1
##           -97.26122   -0.97892  1.76667  -0.01441  14.42040  6.19749
## B2    B3    B4    B5    B6    B7
##     0.36155 11.36319 -1.97553  2.81326 -1.33374  0.76959
## B8    B9
##  -0.70938  1.80938
##
## Degrees of Freedom: 10649 Total (i.e. Null); 10636 Residual
## Null Deviance: 1033000
## Residual Deviance: 959900  AIC: 993300
```

Including Interactions

Similarly the model output does not suggest any major improvement in explaining the spatio-temporal variability in the spread of COVID-19. We may need a different strategy then. Let’s try running a NBM including interaction terms between spatial and temporal terms (i.e. longitude, latitude and day). We can do this by estimating the following model `c_covid19_r ~ (long + lat + day)^2 + lt_illness + .`

```r
# new model specification
eq2 <- c_covid19_r ~ (long + lat + day)^2 + lt_illness + .
# estimate a negative binomial model
nb_m2 <- glm.nb(eq2,
                 data = dplyr::select(reg_df, -ctyu19nm))

nb_m2
```

```r
##
## Call: glm.nb(formula = eq2, data = dplyr::select(reg_df, -ctyu19nm),
## init.theta = 300465.7453, link = log)
##
## Coefficients:
##            (Intercept)      long      lat      day lt_illness       B1
##           -2.642e+02  -7.873e+01  4.936e+00  6.423e-02  1.335e+01  1.145e+01
## B2    B3    B4    B5    B6    B7
##  -7.235e-01  2.132e+01  4.936e+00  6.423e-02 -1.335e+01  1.145e+01
```
## B8 B9 long:lat long:day lat:day
## -5.503e+00 9.127e-01 1.533e+00 8.504e-05 -1.499e-03
##
## Degrees of Freedom: 10649 Total (i.e. Null); 10633 Residual
## Null Deviance: 1033000
## Residual Deviance: 953600 AIC: 987000

This model leads to a better model by returning a slight reduction in the residual deviance and AIC score. Interestingly it also returns highly statistically significant coefficients for the interaction terms between longitude and latitude (long:lat) and latitude and day (lat:day). The former indicates that as we move one degree north and west, the number of new cases tend to increase in 2 cases. The latter indicates UTLAs on the west of England tend to report a lower number of cases as time passes.

You can report the output for all models estimated above by executing (after removing #):

```r
# export_summs(lm_m, poisson_m, qpoisson_m1, nb_m1, nb_m2)
```

### 10.6.2.1 Model Comparision

To compare regression models based on different specifications and assumptions, like those reported above, you may want to consider the cross-validation approach used in Chapter 4 Flows. An easy approach if you would like to get a quick sense of model fit, you can look at the correlation between observed and predicted values of the dependent variable. For our models, we can achieve this by executing:

```r
# computing predictions for all models
lm_cnt <- predict(lm_m)
poisson_cnt <- predict(poisson_m1)
b1_cnt <- predict(nb_m1)
b2_cnt <- predict(nb_m2)
reg_df <- cbind(reg_df, lm_cnt, poisson_cnt, nb1_cnt, nb2_cnt)

# computing correlation coefficients
cormat <- cor(reg_df[, c("c_covid19_r", "lm_cnt", "poisson_cnt", "nb1_cnt", "nb2_cnt")],
               use="complete.obs",
               method="pearson")

# significance test
sig1 <- corrplot::cor.mtest(reg_df[, c("c_covid19_r", "lm_cnt", "poisson_cnt", "nb1_cnt", "nb2_cnt")],
                            conf.level = .95)

# create a correlogram
corrplot::corrplot.mixed(cormat,
                        number.cex = 1)
None of the models does a great job at predicting the observed count of new COVID-19 cases. They display correlation coefficients between 0.23 and 0.24 and high degree of correlation between them. Part of the assignment will be finding ways to improve this initial models. They should just be considered as a starting point.

There are a potentially few easy ways to make some considerable improvement. * **Option 1** Remove all zeros from the dependent variable `c_covid19_r`. They are likely to be affecting the ability of the model to predict positive values which are of main interest if we want to understand the spatio-temporal patterns of the outbreak. * **Option 2** Remove all zeros from the dependent variable and additionally use its log for the regression model. * **Option 3** Include more explanatory variables. Look for factors which can explain the spatial-temporal variability of the current COVID-19 outbreak. Look for hypotheses / anecdotal evidence from the newspapers and social media. * **Option 4** Check for collinearity. Collinearity is likely to be an issue given the way basis functions are created. Checking for collinearity of course will not improve the fit of the existing model but it is important to remove collinear terms if statistical inference is a key goal - which in this case is. Over to you now!
10.7 Questions

We will continue to use the COVID-19 dataset. Please see Chapter 11 for details on the data.

```r
sdf <- st_read("data/assignment_2_covid/covid19_eng.gpkg")
```

## Reading layer `covid19_eng' from data source `/home/jovyan/work/data/assignment_2_covid/covid19_eng.gpkg' using driver `GPKG'
## Simple feature collection with 149 features and 507 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox: xmin: 134112.4 ymin: 11429.67 xmax: 655653.8 ymax: 657536
## projected CRS: OSGB 1936 / British National Grid

Using these data, you are required to address the following challenges:

- Create a spatio-temporal visualisation.
- Fit a ST model to assess changes over space and time.

Analyse and discuss:

1. Do the results for your GWR and ST results differ: How do regression coefficients vary across space? Is this variation statistically significant?
2. Does the ST model indicate significant variations over time? How?
Chapter 11

Data Sets

library(sf)

11.1 Madrid AirBnb

This dataset contains a pre-processed set of properties advertised on the AirBnb website within the region of Madrid (Spain), together with house characteristics.

Availability

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

```r
path <- "data/assignment_1_madrid/madrid_abb.gpkg"
```

```
## Reading layer `madrid_abb' from data source `/home/jovyan/work/data/assignment_1_madrid/madrid_abb.gpkg' using driver `GPKG'
## Simple feature collection with 18399 features and 16 fields
## geometry type: POINT
## dimension: XY
## geographic CRS: WGS 84
```

Variables

For each of the 17 properties, the following characteristics are available:

- **price**: [string] Price with currency
- **price_usd**: [int] Price expressed in USD
- **log1pm_price_usd**: [float] Log of the price
• **accommodates**: [integer] Number of people the property accommodates
• **bathrooms**: [integer] Number of bathrooms
• **bedrooms**: [integer] Number of bedrooms
• **beds**: [integer] Number of beds
• **neighbourhood**: [string] Name of the neighbourhood the property is located in
• **room_type**: [string] Type of room offered (shared, private, entire home, hotel room)
• **property_type**: [string] Type of property advertised (apartment, house, hut, etc.)
• **WiFi**: [binary] Takes 1 if the property has WiFi, 0 otherwise
• **Coffee**: [binary] Takes 1 if the property has a coffee maker, 0 otherwise
• **Gym**: [binary] Takes 1 if the property has access to a gym, 0 otherwise
• **Parking**: [binary] Takes 1 if the property offers parking, 0 otherwise
• **km_to_retiro**: [float] Euclidean distance from the property to the El Retiro park
• **geom**: [geometry] Point geometry

**Projection**

The location of each property is stored as point geometries and expressed in longitude and latitude coordinates:

```sql
ST_CRS(db)
```

```wkt
GEODESIC ["WGS 84",
    DATUM ["World Geodetic System 1984",
        ELLIPSOID ["WGS 84",6378137,298.257223563,
            LENGTHUNIT ["metre",1]]],
    PRIMEM ["Greenwich",0],
    ANGLEUNIT ["degree",0.0174532925199433],
    CS [ellipsoidal,2],
    AXIS ["geodetic longitude (Lon)",east,
        ORDER [2],
        ANGLEUNIT ["degree",0.0174532925199433]],
    AXIS ["geodetic latitude (Lat)",north,
        ORDER [1],
        ANGLEUNIT ["degree",0.0174532925199433]],
    USAGE ["unknown", "World", BBOX [-180, -90, 180, 90], ID ["EPSG", 4326]]
```
Source & Pre-processing

The data are sourced from Inside Airbnb. A Jupyter notebook in Python (available at `data/assignment_1_madrid/clean_data.ipynb`) details the process from the original file available from source to the data in `madrid_abb.gpkg`.

11.2 England COVID-19

This dataset contains:

- daily COVID-19 confirmed cases from 1st January, 2020 to 2nd February, 2021 from the GOV.UK dashboard;
- resident population characteristics from the 2011 census, available from the Office of National Statistics; and,
- 2019 Index of Multiple Deprivation (IMD) data from GOV.UK and published by the Ministry of Housing, Communities & Local Government.

The data are at the Upper Tier Local Authority District (UTLAD) level - also known as Counties and Unitary Authorities.

Availability

The dataset is stored on a Geopackage:

```r
sdf <- st_read("data/assignment_2_covid/covid19_eng.gpkg")
```

```r
## Reading layer `covid19_eng' from data source `/home/jovyan/work/data/assignment_2_covid/covid19_eng.gpkg' using driver `GPKG'
## Simple feature collection with 149 features and 507 fields
## geometry type: MULTIPOLYGON
## dimension: XY
## bbox: xmin: 134112.4 ymin: 11429.67 xmax: 655653.8 ymax: 657536
## projected CRS: OSGB 1936 / British National Grid
```

Variables

The data set contains 508 variables:

- `objectid`: [integer] unit identifier
- `ctyu19cd`: [integer] Upper Tier Local Authority District (or Counties and Unitary Authorities) identifier
- `ctyu19nm`: [character] Upper Tier Local Authority District (or Counties and Unitary Authorities) name
- `Region`: [character] Region name
- `long`: [numeric] longitude
- `lat`: [numeric] latitude
- `st_area_sha`: [numeric] area in hectare
• X2020.01.31 to X2021.02.05: [numeric] Daily COVID-19 cases from 31st January, 2020 to 5th February, 2021
• IMD...Average.score - IMD.2019...Local.concentration: [numeric] IMD indicators - for details see File 11: upper-tier local authority summaries.
• Residents: [numeric] Total resident population
• Households: [numeric] Total households
• Dwellings: [numeric] Total dwellings
• Household_Spaces: [numeric] Total household spaces
• Aged_16plus to Other_industry: [numeric] comprise 114 variables relating to various population and household attributes of the resident population. A description of all these variables can be found here
• geom: [geometry] Point geometry

Projection

Details of the coordinate reference system:

```r
st_crs(sdf)
```

```r
## Coordinate Reference System:
## User input: OSGB 1936 / British National Grid
## wkt:
## PROJCRS["OSGB 1936 / British National Grid",
##   BASEGEOGCRS["OSGB 1936",
##     DATUM["OSGB 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,
- LengthUnit: "metre", 1,
- Axis: "(N)", north,
- LengthUnit: "metre", 1,

## USAGE
- Scope: unknown
- Area: "UK - Britain and UKCS 49°46'N to 61°01'N, 7°33'W to 3°33'E"
- BBox: [49.75, -9.2, 61.14, 2.88]
- ID: "EPSG", 27700]
Bibliography


