Imperial College London

MSc Individual Project

Imperial College London

Department of Public Health

Quantifying the Benefits of Transactional Data in Public Health Research

Author:

CID: 01128684

Word count:

10908 words (excluding table of contents, glossary, abstract, figures...)

Supervisors:

anonymous1

anonymous2

Summer 2020

Acknowledgements

First of all I wanted to thank supervisor *anonymous1* for always being available to help and provide timely answers to all my questions despite its very busy schedule. I also wanted to thank supervisor *anonymous2* for helping me refine my project objectives, for providing timely revision to my early drafts and for showing me when it was a good time to stop writing code and start writing this report.

Abstract

Our behaviour conditions our health. The products and services we consume on a daily basis have a direct impact in our health but are often not measured in public health research. Most transactions we perform nowadays are recorded in the form of transactional data. In this work we present transactional data as a candidate tool to measure behaviour in public health research. We hypothesise that ecological health variables related to health such as life expectancy can better be predicted if we use transactional data and that structure in the transactional data implicitly encodes health information. We also introduce the concept of behavioural resolution in transactional data and we explore whether greater resolution leads to better predictions. We link data from Visa and Office for National Statistics (ONS) from England and Wales in 2019 to study the associations between transactional data and health regional metrics. We also apply unsupervised learning methods to card-level features to explore the structure of the data and whether it relates to health at the ecological level.

We show that Visa data alone or Visa data in conjunction with ONS data can improve predictive performance over that obtained with ONS data only. In most cases greater behavioural resolution leads to better predictions. Card-level transactional information fails to encode regional health information. We wish to catalyse individual data linkage with transactional data in public health research to overcome some of the limitations we have faced in this study and for better monitoring and understanding of individual behaviour in relationship to health.

Glossary

acquirer Bank that issued the merchant's card. 9, 10, 41

AVS Address Verification System, Visa internal tool that communicates an address provided by a consumer to the consumer's issuer to verify online transactions. 10, 12, 14

cardholder individual sending the value of a transaction. 9

funding source Type of bank account linked to the card, may be debit, credit, prepaid, charged or deferred debit/. 16, 17, 41

GDP Gross Domestic Product, quantity to measure the economic output of a region. 12

global health metrics Measures which attempt to capture the nature of health status and, indeed, the whole range of physical, mental and social functioning [1]. 7

issuer Bank that issued the card-holder's card. 9, 10, 16, 41

LAD Local authority district, kind of geographical region delimitation in the United Kingdom. 4, 5, 14–18, 23, 37–39

LE Life expectancy, number of years and individual is expected to live based on the mortality rate from the area they live in. 13, 19

MCG Merchant category group, Visa internal encoding for 28 different merchant categories (Transport, tourism, groceries...). 12, 16, 17, 20, 36

merchant business or card-holder receiving the value of a transaction. 9, 10

NSPL National Postcode Lookup Table updated several times every year by the ONS that allow mapping of postcodes to postal sector, to local authorities, all the way up to regions. 14, 15

PAN Personal account number: card-level identifier, unique to every single card. 4, 14–16, 33, 37, 38, 42

postal sector Postal sectors in the UK are the second largest postal geography delimitation after postal units. Their corresponding codes contains between 4 and 6 alphanumeric characters including one space.. 12, 14, 15

postcode Postcodes in the UK are made of alphanumeric characters of a length between 6 and 8 characters including one space. 14, 15

product platform Indicates whether the corresponding card is a commercial, business or corporate card. Business card refer to small businesses whereas corporate cards form part of scheme of cards that may distributed to employees of a certain corporation.. 16, 17

region England has 9 regions: North West, North East, West Midlands, East Midlands, Yorkshire and the Humber, East of England, London, South East and South West. Wales is considered as a single region

. 33

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our wealth and income condition our expenditure patterns: the food we consume, the clothes we wear, how often we travel, whether we use public transport, etc. These in turn effect our health. Traditionally, public health researchers have measured behaviour through surveys which are subject to many sorts of biases. In this work we present transactional data, data recorded for every transaction performed trough electronic payment methods; as a candidate tool to measure behaviour in public health research. We hypothesise that ecological health variables related to health such as life expectancy can better be predicted if we use transactional data and that structure in the transactional data implicitly encodes health information. We also introduce the concept of behavioural resolution in transactional data and we explore whether greater resolution leads to better predictions. In this study, we will be performing an ecological study using England and Wales local authorities as the unit of study in the year 2019. We will perform data linkage between Visa and Office for National Statistics (ONS) data (from 7 different sources).

We show that Visa data alone or Visa data in conjunction with ONS data can improve predictive performance over that obtained with ONS data only. We also observe than in most cases greater behavioural resolution leads to better predictions. Finally, we observe through the use of dimensionality reduction techniques that card-level transactional information fails to encode regional health information. With this study we wish to catalyse individual data linkage with transactional data in public health research to overcome some of the limitations we have faced in this study and for better monitoring and understanding of individual behaviour in relationship to health.

1.1 Background

1.1.1 Measuring health and wealth

First of all, we need to define these two concepts as well as present how these can be measured. The World Health Organisation (WHO) defines health as 'a state of complete physical, mental and social well-being and not merely the absence of disease or infirmity.'[2]. Additionally, health is stated to have several components: social, physical and psychological [1]. Social health relates to the quality of the relationships an individual has. Physical health relates to the state and abilities of an individuals' body: normal functioning of organs, absence of physical disease, ability to exercise, etc. Psychological health relates to self-worth, stress and positive and negative feelings. Given the multi-dimensionality of health, health can be measured through several metrics which are goal and context dependent. This review [1], covers the multiple metrics that have been developed over the years and the purposes they serve. In this study we will be focusing mainly on global health metrics and some survey-based psychological metrics.

The definition of wealth is 'a large amount of money, property, or valuable possessions that a person or a country owns'[3]. In this regard the wealth level of a certain individual increases with their income and shrinks with their expenditure. Additionally, individuals can see their wealth change through increases or decreases in the valuation of the properties and stocks they own.

Below we will summarise the current understanding of the relationship between health and wealth, though as outlined later, we will study wealth in a different light. At the macroscopic level, some economists debate whether the link between health and wealth is causal or if health and wealth are two distinct processes [4]. On the other hand, at the microscopic level, researchers have found associations between financial stress and inflammation marker in a sample of middle-aged US males [5]. The debate is further expanded by one side arguing that poverty influences health and others arguing that inequality influences health [6]. Though some issues are still debated, some facts are known. According to the Marmot review, in England, people that live in the poorest neighbourhoods have in an average a life expectancy seven years shorter than those in the richest neighbourhoods [6][7]. Similar results are found for infant mortality, mental and physical health and other health metrics.

The stance taken in this study is that an individual's wealth conditions their expenditure behaviour which in turn affects their health.

1.1.2 Measuring behaviour in public health research

Tobacco consumption, poor diet & physical inactivity and alcohol consumption were determined to be the top 3 causes of death in the US in 1990 and 2000 [8] (38.2% of all deaths in 2000). A different study also found these factors to be major contributors to early age mortality [9]. Traditional public health research has devised methods to measure such behavioural factors and we will briefly review them below. We will first explore the temporal aspects of these studies and then how behaviour is actually measured.

All studies, either interventional or observational studies benefit from measuring past or present behaviour as this is often either the subject of study (outcome) or accounts as a confounding effect (exposure). Cohort studies pick a group of outcome-free individuals and follow-up their health until the incidence of particular health outcomes [10]. These studies can either be prospective or retrospective. The latter tend to be short as they become more expensive and less feasible the longer the follow-up period lasts. Additionally, costs go up with the frequency at which the researchers wish to measure exposure. On the other hand, retrospective cohort studies allow researchers to compare past exposure between two groups that have obtained two different outcome (e.g. diabetes type II diagnosis). These represent an inexpensive alternative to long prospective cohort studies while also removing the costs of follow-up. [10]. Though given the fact that exposure tracking in retrospective studies is by definition unplanned, available data is often sparse or absent [10]. Other studies such as cross-sectional studies may also measure past or present exposure though would not follow up the health status of the individuals in the study.

Non-behavioural exposure can often be measured objectively (blood marker levels, imaging diagnosis...) while behavioural exposure is often measured through surveys. The media through which these surveys are completed can be: face-to-face, telephone, mail or web-based [11]. Mail and web-based surveys are often self-administered whereas face-to-face and telephone surveys tend to be completed under the supervision of another individual. The dominant media for the last few decades has been telephone based survey such as the Behavioral Risk Factors Surveillance System (BRFSS) by the Centre for Disease Control and Prevention (CDC)[12]. The rapid shift away from landline phone usage in the decade of the 2000 caused a threat to the BRFSS and other similar surveys [11][12]. The design of these surveys must now incorporate other media such as mail and web-based surveys for which the methodology is not as mature [11][12].

Lastly, surveys suffer from several forms of bias and sources of error. Non-response bias occurs when there exists systematic differences in the rate of participation that is related to the factor(s) being measured [13]. Selection bias occurs when the probability of being eligible for a study depends on some characteristic related to the factor(s) being measured [14]. Other biases occur because of the surveyed individuals depiction of their own past exposure. Individuals may forget or have a distorted depiction of past exposure (recall bias)[15], or they deliberately change the response on their past exposure because of what is believed to be non-socially acceptable (social desirability bias)[16] or they may even fail to reveal some information for similar reasons (reporting bias) [17].

1.1.3 Transactional data as a better tool to measure behaviour

As previously stated, we will assume wealth translates to daily behaviour. These daily behaviours are in part translated to what we consume and the products we purchase. In the event a payment for these activities makes use of an electronic payment card, this data (transactional data) is recorded. Such data is abundant in today's world. Banks, supermarket and card issuers, collect and store this data routinely. However due to raising concerns on data privacy, this data has not yet reached the public health research realm [18]. Transactional data enables the monitoring of consumer behaviour through the recording of the transactions being made through electronic payments means. This has a range of benefits. The first one is that it allows a very high frequency sampling of behaviour at virtually no cost, given that the data is processed and stored regardless. The second one is the scale of the operation at also no extra cost. Given a public health study that uses transactional data has not been conducted yet it is hard to assess whether consumer behaviour may change if the participants were to know the data would be used in health studies. Evidently these methods also suffer from other source of biases. Does payment card usage vary across age groups? Across ethnic or social groups? Does the nature of the transactions affect the form of payment (card vs cash)? These are important questions that need to be answered when using transactional data for public health research. Card penetration research to answer these questions exists, though it is often not publicly available as it mostly used by banks and other financial institutions. Skatova et. al performed a preliminary study, where they outlined the main concerns of individuals when it comes to the usage of their transactional data for public health research: "importance of ensuring participants have access to appropriate information, control over their data, and trust in the organisation." [18][19].

1.1.4 VisaNet data specifications

Visa specific terms will be mentioned throughout this study and will all be covered in the glossary.

What is Visa and how are transactions processed?

Visa is a network-payment company, one which enable the validity of a transaction between two parties such as fraud detection to ensure the fair consented exchange of money between these two parties. The parties in a single transaction are the consumer or cardholder, the issuer (the bank that issued the card), the merchant (the business selling something) and the acquirer (merchant's bank) and finally VisaNet which connects all the previous parties together. The three stages any transaction go trough are authorisation, clearing and settlement. A diagram illustrating this process is available in figure 1.1.

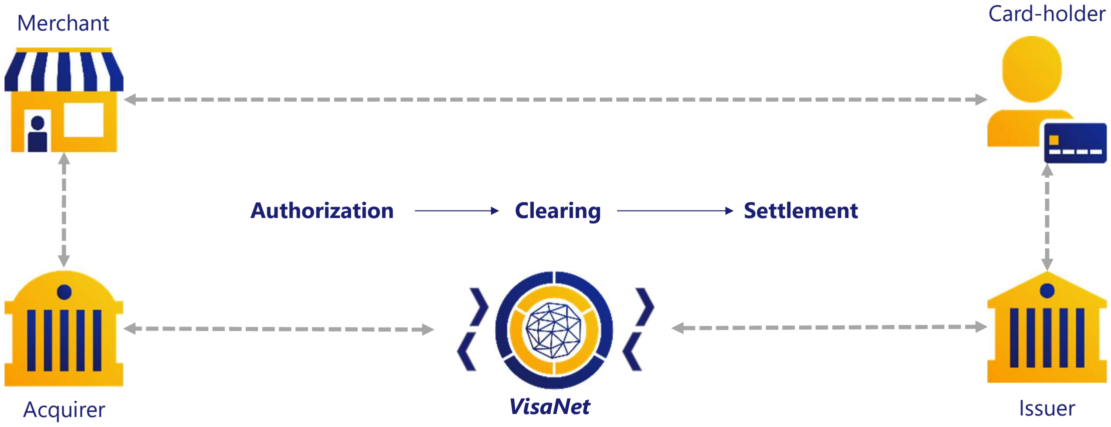


Figure 1.1: Different parties involved in the approval of a single transaction. *Modified from Visa's internal educational material*

The authorisation stage occurs when the cardholder taps, swipes or insert its card at a terminal (either physical or virtual). Next, information about the current time, location and card information is sent to the acquirer which will process the information and transfer it to VisaNet. Here VisaNet will verify the transaction characteristics to ensure the transaction is not fraudulent and will produce a fraud score. This information will be passed to the issuer, which will decide whether the transaction is valid. The issuer will check that the account balance has sufficient funds and hold these funds for the transaction. All this information is now passed back through VisaNet, the acquirer and finally the terminal at the merchant side. If all the verification steps were successful the transaction is approved. This process takes less than a second, though no money has been transferred between any of the accounts at this point. The second stage, clearing, is when the acquirer sends the final amount to be paid to it to VisaNet which appends this data to later send it in batches to all the issuers. The issuers then proceeds to move the money it held out from the account balance of the card-holder's account. Finally, in the settlement stage, Visa sends the net balance statement to all issuers and acquirers, and funds are finally moved between these institutions [20]. More information can be read in Visa's official website [21] and Visa's official communication channels [22][23].

VisaNet Global and UK statistics

As per Visa's fiscal annual report from 2019, 183.3 billion transactions were processed through VisaNet globally accounting for a volume of $11.6 trillions[24]. Additionally, a total of 3.4 billion cards were registered in the VisaNet system in 2019. To put these numbers into perspective, *Mastercard*, Visa's main competitor, had a transaction volume of $5.9 trillion dollars and transaction count of 103 billion transactions over 2018 [24].

In the UK, on which this study is focused, Visa dominates the debit card market, accounting for 96.1%(94.5M cards) of the issued debit cards in the country[25]. On the other hand Visa issues 36.9% of the UK's credit and charged cards (24M cards), whilst Mastercard's issues 57.3% of it (37.2M cards). It is worth noting that the UK is "debit-dominated" market, where 60.2% of the transactions in 2019 was performed with debit cards, and this number is expected to grow [25]. Finally card transactions in the UK account for 54.3% of the transaction volume whereas cash, bank transfers and direct debits account for 22.6%, 11.3% and 11.1% of the transaction volume respectively. This highlights that even though Visa is the leading payment network company in the UK and the UK's population is thoroughly banked, VisaNet does not 'see' all the transactions that happen in the UK.

Misconceptions about Visa data

Visa is not a bank, hence it does not store the cardholder's personal information nor can it look at the cardholder account balance. Visa, as explained previously, is a payment network system hence it can only see the information regarding the transactions being processed. Though VisaNet can obtain some of this information under certain circumstances, either because of the nature of a transaction or because of the use of transactional data to infer other information. As an example, Visa does not directly know the cardholder address nor even postcode, though it can obtain it the following way. When an online transaction occurs the card holder needs to included their billing postcode to confirm their identity, in this situation Visa can see this information to later send it to the cardholder's issuer. This information is stored in the Address Verification System (AVS). In turn, Visa can also build models to predict the district of a cardholder based on the transaction locations of a given card. Information from such models was used in this study.

1.1.5 An Ecological study before individual studies

No study exists using transactional data in the context of public health. Visa data and individual medical records cannot as of today be linked. Because of this we are limited today to perform either an ecological study or a study with no outcome signal (unsupervised learning). An ecological study is one that uses regions or groups of individuals as the unit of analysis instead of an individual person [26]. These studies are cheap as they can make use of readily available data but suffer major pitfalls. The most common pitfall is the ecological fallacy, which occurs when the association at the group level are taken as true at the individual level [26]. Additionally, the loss of granularity means that intra-group variance is not quantified in the analysis.

Regardless of the above mentioned limitations, data linkage at the regional level represents the only way to link transactional data to public health metrics, hence this is how we will proceed in this study. In particular, we will be linking Visa data with data from the Office for National Statistics (ONS) from the year 2019 in England and Wales at the local authority level. Data from only 2019 was chosen because of two reasons. First this subset is the latest pre-COVID set of data. COVID19 has greatly affected expenditure patterns which would interfere with the purpose of our study. Secondly the data being from a single calendar year makes it simpler to link the data with publicly available data.

We will attempt at studying the structure of the transactional data at the card-level through unsupervised learning techniques such as dimensionality reduction and clustering.

1.2 Objectives

- Does transactional data improve performance of public health metric predictions? Does higher-resolution of transactional data further improve such performance?
- Does transactional data encode health-related information? Can unsupervised learning methods find clusters that relate to public health metrics?

Chapter 2

Methods

2.1 Data sources

2.1.1 Visa

We focus on the universe of authorised transactions (from the authorisation processing stage - see section 1.1.4) that happened in 2019 for individuals whose residence was either England or Wales. We used the authorisation table to be able to count the number of declined transactions. The cardholder postcode was either obtained through the AVS or obtained from an inference model (Visa internal) that predicts the postal sector of an individual based on their expenditure in face-to-face transactions. I was not involved in the development of this model, it may be regarded as another internal data source.

The authorisation table contains information about every attempted transaction, including those that are declined due to lack of funds, wrong verification or because of a potential attempt of fraud. A list with all the information per transaction in the authorisation table can be found in A.1. It is important to introduce here the global merchant repository, merchant category code and merchant category group (MCG). The global merchant repository is a database kept internally at Visa to map payment terminals to their corresponding merchants. The merchant category code and merchant category groups are identification numbers that allow to categorise the market segment a certain enterprise corresponds to. The list of 28 MCGs is available in A.1.

2.1.2 ONS

In addition to the Visa data, we extracted data from the Office for National Statistics (ONS) at the local authority level. More precisely, we extracted seven tables with the following information: life expectancy, number of suicides, wellbeing metrics (based on wellbeing surveys), population (number of people per age bin – also stratified by gender), gross domestic product (GDP), unemployment and work. The information about the downloaded data as well as with the dates at which the data was downloaded is available in table 2.1. The download links for data is available in table A.1.

Explanation of each source and how it was obtained:

Each data source comes with its own Quality & Methodology Information (QMI) section. Below we will be explaining the main features that are necessary to understand the data sources and the corresponding estimated statistics.

Life expectancy

This dataset includes the life expectancy (LE), healthy life expectancy (HLE) and disability-free life expectancy (DFLE) (at birth and at 65 years of age) for all the local authorities in the

Dataset	**Years**	**Information**	**Date Accessed**
Life expectancy	2013-2017	Life expectancy	May 28th
Suicides 1	2002-2017	Suicides time-series	May 28th
Suicides 2	2018	Suicides 2018	May 28th
Wellbeing	2011-2019	Wellbeing survey data	May 28th
Work	2019	Earnings and hours worked	May 28th
GDP	1998-2018	Regional GDP	May 28th
Population 1	2012-2017	Population and age	May 28th
Population 2	2018-2019	Population and age	June 22nd
Unemployment	1996-2019	Unemployment rate	June 25th

Table 2.1: Information and time coverage of the ONS tables

United Kingdom (UK) for the intervals 2013-15, 2014-16 and 2015-17 [27]. The estimates are stratified by gender. LE is defined as the number of years (on average) that a person in a given area is expected to live given the mortality rates in that area. HLE is number of expected years to be lived in "good" health. DFLE is the number of expected years to be lived without restrictions cause by a severe physical or mental health condition. The data is provided in 3-year intervals as the data is calculated using rolling averages to ensure robustness [28]. We only used the normal LE stratified by gender at birth and at 65, as HLE and DFLE was missing for many LADs (see ONS cleaning notebook A.5). We obtained gender-less estimates of the LE at birth and at 65 by calculating a population weighted average from the male and female LE values:

$$LE_{genderless} = \frac{LE_{male} \cdot Male_{pop} + LE_{Female} \cdot Female_{pop}}{Total_{pop}} \tag{2.1}$$

Suicides

Suicides data was downloaded for the years 2002 to 2018. Suicides are defined as "deaths with an underlying cause of intentional self-harm (ages 10 years and over) and deaths with an underlying cause of event of undetermined intent (ages 15 years and over)." [29]. Suicide records in England & Wales come from the ONS Deaths Registrations Database. The cause of death in the ONS is registered as per the International Classification of Diseases, 10th Revision (ICD-10). We only used the suicide count per local authority though more information is available such as age-specific suicide rate and suicide method [30][31].

Wellbeing

Every year the ONS conducts the Annual Population Survey(APS). In this survey, individuals are asked amongst other things to self-report on their personal wellbeing. The survey sample size for 2018 included approximately 150 thousand people[32]. Measuring wellbeing is done through four questions relating to life satisfaction, worthwhile, happiness and anxiety (see table 2.2). For each question 0 means "not at all" and 10 means "completely". The aggregate data comes as average score values (out of 10) over the local authority and as population proportions per score bins, with 4 possible score bins: Poor, Fair, Good, Very Good [33]. In this study we only made use of average score values as they were easier to interpret and incorporate into our data. According to the ONS, the fact that the wellbeing measures are self-reported allows to "take account of what matters to people" instead of trying to find objective wellbeing metrics as it was done in previous years[32].

Population

Every ten years, the UK government perform its census where it aims to collect population data across the whole UK. This is done by raising awareness about the census and sending out census questionnaires to everyone in the UK [34]. In between census, the ONS calculates population estimates based on the previous census data while incorporating data from birth, death and migrations records [35]. This data includes population count broken down by age (in 1-year bins) and gender. Individuals over 90 years of age are put together in a 90+ bin.

GDP

The GDP measures the total economic activity of a country. Three main ways of calculating the GDP of a country exist: the output, the input and the expenditure approach [36]. The main

Measure	Question
Life satisfaction	*Overall, how satisfied are you with your life nowadays?*
Worthwhile	*Overall, to what extent do you feel that the things you do in your life are worthwhile?*
Happiness	*Overall, how happy did you feel yesterday?*
Anxiety	*On a scale where 0 is "not at all anxious" and 10 is "completely anxious", overall, how anxious did you feel yesterday?*

Table 2.2: ONS APS wellbeing questions [32]

approach used by the ONS to calculate the national GDP is the output approach which calculates the difference between the output from the products and services exported and the cost of creating those products and services[37]. The GDP data provided at the LAD level is calculate based on the regional gross value added [38] which in turn is calculated based on the UK National Accounts(The Blue Book). All these data sources are compiled by the ONS.[39].

Unemployment

The unemployment rate is not what most people think it is. The unemployment rate is not the proportion of people of working age that are unemployed. Instead, the standard definition of this metric is the proportion of the population that has lost their employment in the last 4 weeks and that is ready to start a job in the next two weeks [40]. The data we used was sourced from an ONS model-based approach to estimate unemployment [41]

Work

The Annual Survey for Hours and Earnings (ASHE) is yet another ONS survey which collects information regarding hours worked and salary [42]. It records variables such as annual, monthly, weekly and hourly pay, workplace (own residence or office), working patterns (full or part-time) and other statistics [43]. Out of this dataset we only selected the information regarding hourly pay stratified by gender and also aggregated (gender-less).

Finally, we downloaded the National Statistics Postcode Lookup (NSPL) table from the ONS Open Geographical Portal [44]. This was used to be able to match postcodes or postal sector codes to LAD codes.

2.2 Data wrangling & cleaning

2.2.1 Visa

A diagram visually describing the preprocessing of the data is available in figure 2.1.

For ease of understanding I will describe the preprocessing steps by process ID:

Process 1, Left join the UK authorised table with the table containing the inferred postal sector codes, further more filter all transactions performed outside 2019 and those that did not have any postcode information available (from either of the former sources)

Process 2, For each PAN in the table obtained from Process 1:

1. Remove those entries that satisfy either of the following conditions: entries where the code from the inference model is shorter than 3 characters or longer than 5. The model was meant to predict postal sector codes, which by definition in the UK must be of the length stated [45] OR where the AVS postcode contains no letters.
2. If the AVS postcode is not available pick the model postal sector

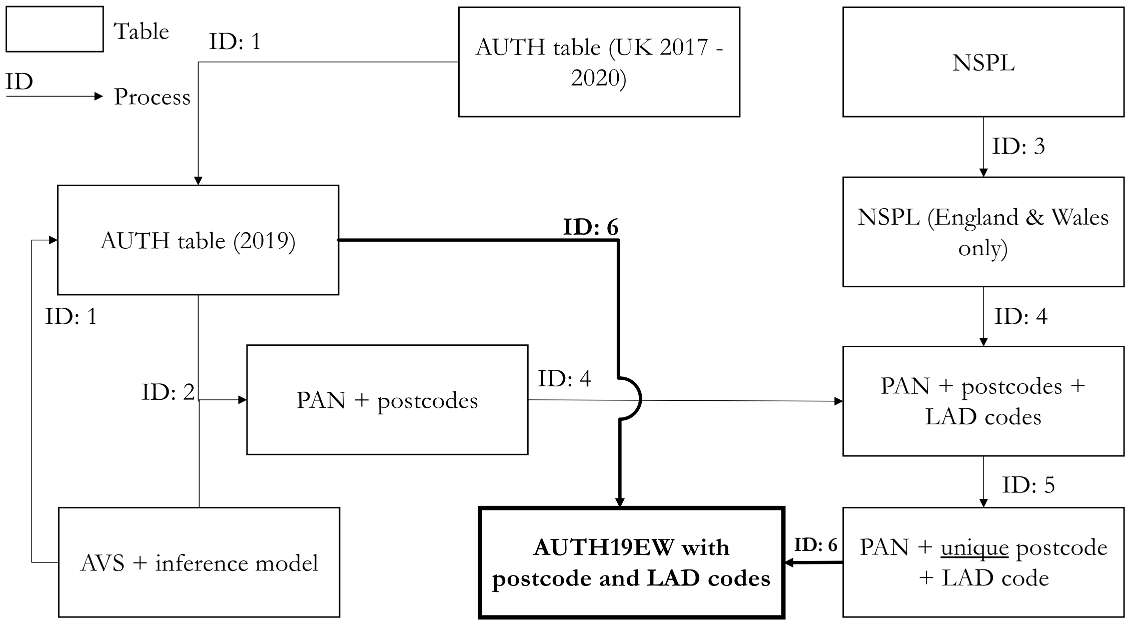


Table
ID
Process
ID: 1
AUTH table (UK 2017 - 2020)
NSPL
ID: 3
AUTH table (2019)
ID: 6
NSPL (England & Wales only)
ID: 1
ID: 4
ID: 2
PAN + postcodes
ID: 4
PAN + postcodes + LAD codes
ID: 5
AVS + inference model
AUTH19EW with postcode and LAD codes
ID: 6
PAN + unique postcode + LAD code

Figure 2.1: Visual description of the extraction from the raw Visa data

3. Remove leading numbers, or special characters such spaces, asterisks(*), hyphen (-), dots (.), back or forward slashes (/) or (\), pluses (+) or apostrophes (')
4. Count the number of transactions per pan per stored postal sector code postcode (to be used later) and keep only distinct rows

Process 3, Filter out entries from outside England or Wales (Country codes: E92000001 AND W92000004 respectively) and deprecated postcodes in the NSPL table

Process 4, For each postcode available in the reduced NSPL table, remove the two last characters to obtain the postal sector, to enable matching with inference model postal sector codes. Finally merge the postal sector codes or postcodes obtained in Process 2 to get a LAD code for every pan.

Process 5, Some PANs have several postcodes/postal sector codes hence for every PAN assign the LAD code where most transactions occurred.

Process 6, Merge the table obtained in Process 1 with that obtained in Process 5 to obtain the final table including transactions that occurred in 2019 of individuals residing in England or Wales with their estimated LAD code.

The original authorisation table (at the top of the diagram) contained 77.7 billion transaction records from over 996.5 million PANs (between January the 1st, 2017 and June 28th 2020). The final table with information about individuals from England and Wales only and for the year 2019 contained 14.5 billion transactions records and 72.1 million unique pans.

2.2.2 ONS

The preprocessing done on the seven tables was very similar, though with some slight occasional differences. For most, the processing consisted of theses steps:

1. Remove redundant (duplicate) columns, rename variables, change variable types

2. If per-interval values were available (instead of per-year) take year as the latest year in the interval

3. Perform time-series imputation, if only a single non missing value was available, perform last-one-carry-forward (LOCF) imputation, otherwise, perform linear interpolation to obtain the missing values.
4. Select only entries for the latest available year (2017 for life expectancy data, 2018 for population, suicides, GDP and 2019 for wellbeing,work and unemployment)
5. Arrange data to a tidy format, where each row corresponds to a single entry and each column corresponds to a feature

After all these steps the data is then merged into a single table containing all the extracted ONS data per LAD. At this stage, this table contained 446 entries and 92 columns (including LAD code and name). The list of available columns at this stage is available in the appendix A.2.

A document describing the cleaning and processing of the ONS data is available in the appendix A.5 or online (for better visualisation) in this link.

2.3 Feature engineering

Two tables stemmed from the raw data described in section 2.2.1. One contains aggregate data at the local authority district (LAD table) level while the other one aggregates data at the card-level (PAN table). The LAD table contains in each row aggregate information for each LAD in England and Wales. The PAN table contains in each row information about a single card number over the course of 2019, also from England and Wales. The steps to obtain each of these tables is outlined below.

2.3.1 PAN-wise table

First we extracted the PAN-wise features, where each row corresponds to features of expenditure of a single PAN over the course of the year 2019. A figure outlining the extraction process can be found in figure 2.2. We extracted 130 features (excluding the PAN number), including transaction-wise statistics and pan-wise statistics. The former includes the number of attempted, approved, declined, card-present, contactless, mobile payment transactions (6 features). This also included the sum, average and standard deviation of the transaction value over the course of the year (3 features) and stratified per merchant category group (MCG) (84 features) as well as the count of transactions per MCG (28 features). Additionally, the pan-wise statistics included the postcode or postal sector code, LAD code an country code attributed to each PAN, the funding source, issuer ID and product platform and the number of months in 2019 where that PAN attempted to transact or successfully transacted (9 features).

Finally only entries with more than 50 transactions in the course of 2019 and that transacted every month were kept, leaving us with 23,566,687 entries (unique PANs)

A summary table describing the statistics of the final PAN table can be found in 3.1. Only 11 of the 129 features were shown as most features were made with predictive performance as a goal instead of interpretability.

2.3.2 LAD-wise table

As shown in figure 2.3, 152 features(excluding the LAD code) were extracted per LAD from the final table obtained in Process 6 explained in section 2.2.1. The statistics we gathered can be separated in three categories: transaction-wise, account-wise, pan-wise. The transaction-wise statistics are the count of attempted, approved, declined, card-present(face to face), contactless, mobile(Apple Pay, Google Pay...) transactions (6 features) as well as the sum of the transaction volume, the average transaction value, and the standard deviation in transaction value (3 features). Additionally, we included the sum, average and standard deviation of transaction value per MCG (84 features) as well as the count of transactions per MCG(28 features). The account-wise statistics were the total number of bank accounts(1 feature), the number of account per bank (19 features -

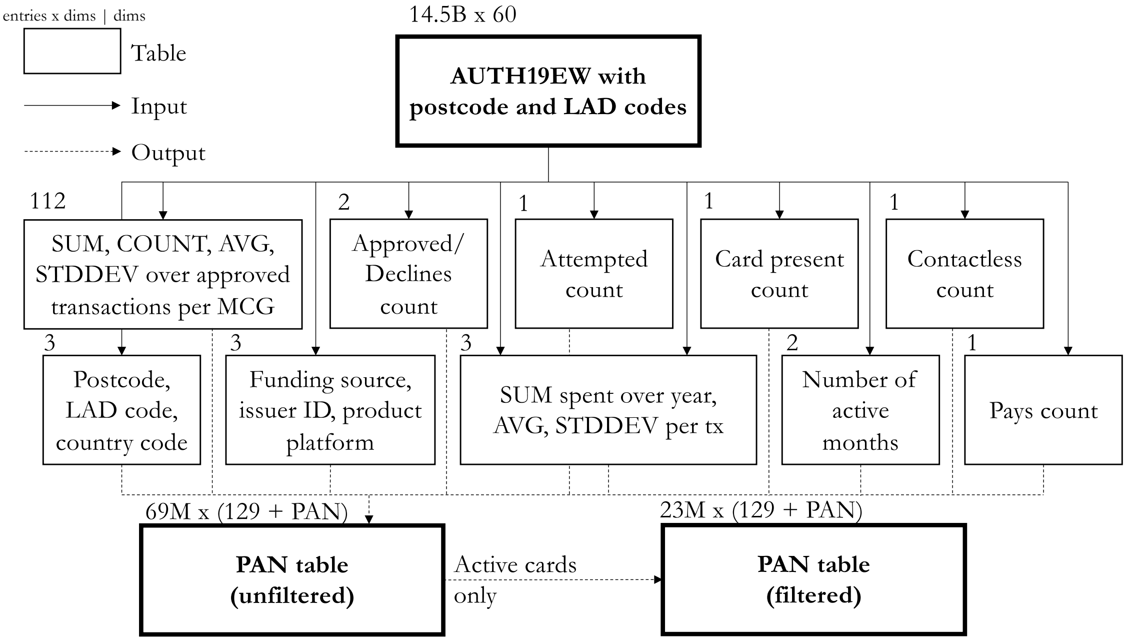


Figure 2.2: Entity relationship diagram to obtain the PAN table

for 19 banks), the number of account per funding source (6 features) and per product platform (3 features). Finally we included the average and standard deviation of the sum spent per PAN over the course of the 2019 year (2 features - pan-wise statistics).

A summary table describing the statistics of the final PAN table can be found in 3.2. Only 13 of the 152 features are shown.

Mid and high-precision LAD features

If we want to use transactional data to measure behaviour of cardholders we can do this at several levels of what we will call behavioural resolution or simply resolution. We can measure the sum spent over a year (for the lowest resolution). Next, we can also see the sum spent per MCG (e.g. Groceries) (mid resolution). Finally we can further explore the sum spent per MCG and per enterprise (e.g Groceries, Tesco or Groceries, Waitrose) (high resolution). In practice, to get the high resolution information we ranked each enterprise in each MCG by the transaction volume in 2019 over England and Wales. To obtain further levels of granularity, we made three distinct tables. In the first only the statistics for the top 5 enterprises per MCG would be calculated where as all the enterprises per MCG that were not in the top 5 would be grouped as others. Then we did the same with only enterprises amongst the top 10 per MCG and top 20 per MCG. We obtained tables including the sum, average and standard deviation of transaction value per MCG for these different tables enterprises as well as the count of transaction per the same categories. We called these tables top5, top 10 and top 20. A diagram explaining this conceptual procedure is available in figure 2.4

2.4 Analytical goals, merging and missing data

2.4.1 Project goals and analytical goals

Definition of outcome variables

Outcome variables were always ONS variables that were related to health (physical or mental) and deprivation (economical or social). The variables we initially selected as outcomes were: the life expectancy estimates (for males, females, overall and at birth or at 65 years of age), the number of suicides, the wellbeing survey data (average scores and range), the unemployment rate and the

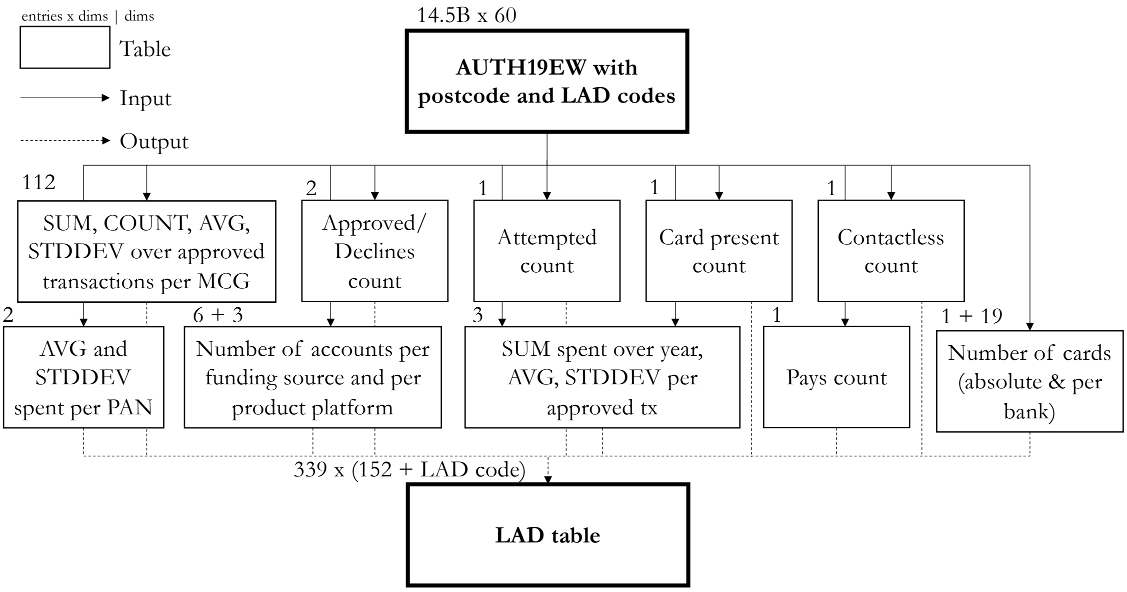

entries x dims | dims
Table
Input
Output
14.5B x 60
AUTH19EW with postcode and LAD codes
112
SUM, COUNT, AVG, STDDEV over approved transactions per MCG
2
Approved/ Declines count
1
Attempted count
1
Card present count
1
Contactless count
2
AVG and STDDEV spent per PAN
6 + 3
Number of accounts per funding source and per product platform
3
SUM spent over year, AVG, STDDEV per approved tx
1
Pays count
1 + 19
Number of cards (absolute & per bank)
339 x (152 + LAD code)
LAD table

Figure 2.3: Entity relationship diagram to obtain the LAD table

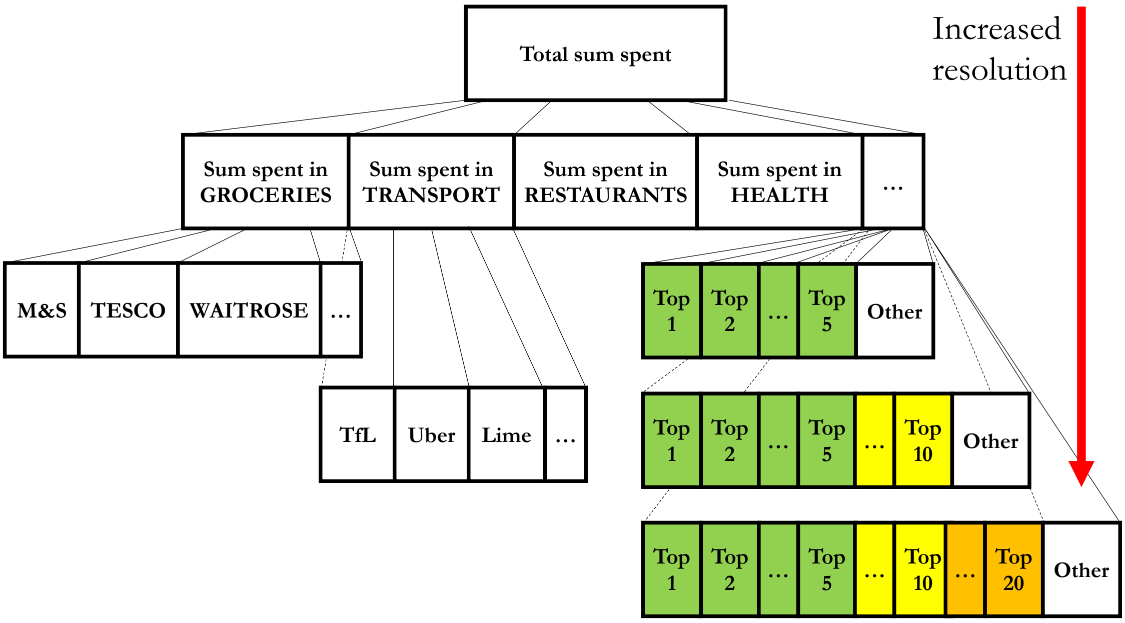


Figure 2.4: Behavioural resolution diagram. *The top level is considered the be the one with the lowest resolution, while resolution increases in downward branches.*

GDP. In practice we only reported results for LE at birth and at 65, number of suicides, average happiness score, unemployment rate and GDP. Additionally, when one LE variables was taken as the outcome all the other LE variables were not used as predictors as these are too correlated to each other. When the outcome was not an LE variable, no LE variables were taken as predictors (effectively resulting in one less predictor).

We did not define explicitly what we considered to be confounding and predictor variables because of two reasons. First, our study is not focused on explainability but rather on predictive performance hence all of our variables were going to be passed through our subsequent models. Second, we made the choice of using models that implement automatic variable selection (later section) hence we did not manually select which variables were deemed relevant.

2.4.2 Data merging

We merged the LAD table presented in section 2.3.2 along with ONS table which left us with 339 entries and 248 columns. We also merged the PAN table with the ONS table based on the LAD code assigned to each PAN.

2.4.3 Missing Data, Imputation & Dropping Outliers

The Visa data did not include any missing values given it was all aggregate data over a year. Time-series imputation on the ONS data could only provide missing values for the 2019 year where data for previous year was available. Unfortunately, some LADs had no data available for any of the recorded years for certain variables. These included five local authorities: Oadby and Wigston, Somerset West and Taunton, Isles of Scilly, City of London and Bournemouth, Christchurch and Poole. A table with what outcome variables these were missing is available at 2.3. Additionally, we removed the columns from the ONS dataset corresponding to the lower and upper confidence intervals of the LE estimates (8 features) and the coefficient of variation estimates of the hourly pay estimates (18 features).

Missing data for these LADs is assumed to depend on factors not measured within the available datasets, hence we assumed we are dealing with a missing not at random mechanism (MNAR). For example, we determined that the fact that the LE values for the City of London were missing did not depend on the City of London's population size or expenditure patterns but rather on some other unmeasured characteristics. Hence we decided to exclude these LADs from the analysis. We performed the imputation of all the variables that were not outcome variables using the package `missRanger` [46], which combines chained random forest with multiple imputation [47][48]. Random forest imputation has been shown to out-perform other imputation algorithms, hence why it was our algorithm of choice [49]. We did not impute any of the outcome variables values as the entries with missing data in any of the outcomes are already been removed from our data. Additionally, we did not use any of the outcome variables values to impute the predictor variables.

At this stage, all the columns were standardised to have a mean of 0 and a standard deviation of 1 and this data was used in all subsequent steps.

2.5 Predictive models and model performance comparison

2.5.1 Datasets

We included 13 different datasets in our experiment. For simplicity we will gives aliases to these tables, we may refer to these tables by their aliases or by the numerical IDs. These included:

1. (ONS) ONS data only
2. (Visa low) Low resolution visa data: transaction-wise (attempted transactions, sum, average, standard deviation of transaction value over year) and pan-wise data (average and standard deviation of expenditure per pan and number of cards)

	Oadby and Wigston	Somerset West and Taunton	Isles of Scilly	City of London	Bournemouth, Christchurch and Poole
Female LE at 65	NM	NM	***M***	***M***	***M***
Female LE at birth	NM	NM	***M***	***M***	***M***
Male LE at 65	NM	NM	***M***	***M***	***M***
Male LE at birth	NM	NM	***M***	***M***	***M***
Number of suicides	NM	NM	NM	NM	NM
Unemployment rate	NM	NM	***M***	***M***	NM
Avg. anxiety score	NM	NM	***M***	***M***	***M***
Avg. happiness score	NM	NM	***M***	***M***	***M***
Avg. life satisfaction score	NM	NM	***M***	***M***	***M***
Avg. worthwhile score	NM	NM	***M***	***M***	***M***
Range anxiety score	***M***	***M***	***M***	***M***	***M***
Range happiness score	***M***	NM	***M***	***M***	***M***
Range life satisfaction score	NM	NM	***M***	***M***	***M***
Range worthwhile score	***M***	NM	***M***	***M***	***M***
GDP	NM	NM	NM	NM	NM

Table 2.3: LAD entries with missing outcome variables. *M stands for Missing, NM stands for Not-Missing*

3. (Visa mid) Mid resolution Visa data: columns in Visa low and transaction-wise statistics per MCG and count of approved, declined, card present, contactless and mobile payments transactions.

4. (Visa high) High resolution Visa data: columns in Visa mid and account-wise statistics (number of accounts per bank, per product platform and per funding source)

5. (ONS + Visa low) 1 and 2

6. (ONS + Visa mid) 1 and 3

7. (ONS + Visa high) 1 and 4

8. (ONS + Visa mid + top5) 6 and the sum, average, standard deviation and count of transactions per MCG per enterprise in the top 5 of each MCG or not in the top 5 (see section 2.3.2)

9. (ONS + Visa mid + top10) 6 and the sum, average, standard deviation and count of transactions per MCG per enterprise in the top 10 of each MCG or not in the top 10 (see section 2.3.2)

10. (ONS + Visa mid + top20) 6 and the sum, average, standard deviation and count of transactions per MCG per enterprise in the top 20 of each MCG or not in the top 20 (see section 2.3.2)

11. (ONS + Visa high + top5) 7 and top 5

12. (ONS + Visa high + top 5 + top10) 11 and top 10

13. (ONS + Visa high + top5 + top10 + top20) 12 and top 20 (all the tables combined)

The dimensionality of each of these tables is available in table 2.4.

Table ID	Components	Dimensionality(rows x columns)
1	ONS	334 x 60
2	Visa low	334 x 7
3	Visa mid	334 x 124
4	Visa high	334 x 152
5	ONS + Visa low	334 x 67
6	ONS + Visa mid	334 x 184
7	ONS + Visa mid + top 5	334 x 349
8	ONS + Visa mid + top10	334 x 1384
9	ONS + Visa mid + top 20	334 x 2464
10	ONS + Visa high	334 x 212
11	ONS + Visa high + top 5	334 x 377
12	ONS + Visa high + top 5 + top 10	334 x 1549
13	ONS + Visa high + top 5 + top 10	334 x 3829

Table 2.4: Overview of the tables used in the model performance comparison experiments.

2.5.2 LASSO, cross validation and subsampling procedures

The code described in this section was applied to all the datasets mentioned in the previous section. Our goal here was to measure the predictive performance of a model at predicting the outcomes defined in section 2.4.1 (one at a time) using the different resolution datasets we have created. To achieve this we need our implementation to include several subtleties. We did not want to manually exclude any variable and additionally the number of columns in our datasets was large (see 2.4). Hence we chose to use LASSO [50] for automatic variable selection (see 2.5.2 for reference). Secondly, and because of the small size of our datasets we split our data in a training and a test set (80:20 split), in order to calibrate our model with the training data and evaluate its performance on the test. Moreover, we performed 3-fold cross-validation on the training set to fine-tune the regularisation parameter λ of the LASSO model. Again, because of the size of our dataset, we decided to implement 100 sub-sampling iterations where at each iteration a different training and test set were chosen. To ensure the same data was being passed in each sub-sampling iteration when changing the outcome variables we set the random seed to be equal to the iteration number in each sub-sampling run. We also set fold IDs for each dataset entry so that the cross-validation datasets were also equivalent when using different outcomes. After cross validation, the λ corresponding to the model with a loss 1 standard error away from the lowest loss was chosen (known as the 1SE rule). Finally, a prediction was obtained with the chosen model and the root-mean-squared error (RMSE or L2 loss) with respect to the test values was calculated and stored.

Algorithm 1: Sub-sampling LASSO implementation

```
Result: RMSE on test set for 100 subsampling iterations
y ← chosenoutcome for subsample in 1 to 100 do
    seed = subsample;
    Shuffle X and y;
    Training set ← set 80% of X and y;
    Test set ← remaining 20% of X and y;
    Split training set in 3 folds;
    Perform cross-validation of LASSO model with training data;
    Pick best $\lambda$ for LASSO based on 1SE-rule;
    Calculate RMSE between predicted value for test X and test y;
end
```

LASSO

The Least-Absolute Shrinkage and Selection Operator (LASSO) was popularised by Tibshirani [50]. LASSO extends the ordinary least squares (OLS) solution to linear regression by imposing a constrain to the absolute value of the regression coefficients β. Mathematically:

$$\begin{aligned}(\hat{\alpha}, \hat{\beta}) = argmin \sum_{i=1}^{N} (y_i - \alpha - \sum_j \beta_j x_{ij})^2 \\ \text{subject to } \sum_j |\beta_j| \leq t\end{aligned} \tag{2.2}$$

where $\hat{\alpha}$ and $\hat{\beta}$ are the estimated regression coefficients, i is the index of the observations and j is the index of the features in the design matrix X and y_i is the outcome of observation i, t is a hyper-parameter, where greater.

By design, this method is able to shrink certain coefficients to 0 which results in better interpretability but also in better generalisation. Another added benefit of LASSO is that the matrix of predictors X does not need to be of full rank [50] which is favourable in our case where certain predictor are likely to be very correlated.

2.6 Unsupervised learning of transactional data

In this section we were interested in seeing if gradients or clusters were formed in the higher dimensional spaces of the transactional data we collected that would relate to some external variable from the ONS. All the unsupervised learning models were trained with Visa data only. After appropriate dimensionality reduction, we plotted the 2D embedding and coloured the points to investigate the occurrence of clusters or gradients of the external label value.

2.6.1 LAD table

We tested several dimension reduction, clustering and visualisation algorithms to investigate the structure of the high-dimensional data. We tested the use of KMeans, t-distributed Stochastic Neighbour Embedding(t-SNE)[51], Negative Matrix Factorisation (NMF), hierarchical agglomerative clustering (HAC), Density-based spatial clustering of applications with noise (DBSCAN) [52], Gaussian mixture Models and Uniform Manifold Approximative Projection (UMAP)[53]. We will only be describing the work with UMAP given that we found it to be the preferred method in terms of ease-of-use, ease-of-understanding and scalability to bigger datasets. We will also describe the PCA and t-SNE methods in details as UMAP is often compared to these across the literature.

For the LAD data UMAP embeddings, after calibration (not shown), we chose 20 as the number of nearest neighbours and otherwise the default settings of the `UMAP` python package [54] as we judged those yielded the clearest visualisations.

2.6.2 PAN-wise clustering

Here we only explored UMAP for dimensionality reduction since this algorithm perform very well for very high dimensional data, as opposed to t-SNE for example. Additionally, we made use of the GPU enabled `cuML` implementation [55] which further enabled us to explore several UMAP hyper parameters (number of neighbours to focus on, minimal distance between embedded points in the low dimensional space). Due to computational limitations we did not use the 23 million observations from our dataset but instead used 1 and 5 million randomly sampled points from the original dataset to perform the embeddings. Greater values for the number of nearest neighbours hyper-parameter also increased computational cost. Hence we focused mostly on the embedding of the 1 million data points to be able to test a wider range of nearest neighbours values. We tested all possible combinations of the nearest neighbour parameter and the spread parameter for the following values: number of neighbours of 10, 30, 75, 100, 200 or 300 and spread of 0.1, 0.5, 1, 10 and 50. The spread parameter controls how spread out the points in the low dimensional space will be with lower values resulting in more clumped embeddings.

As opposed to the LAD table in this scenario we did not possess any external data to label our transactional data. Instead we assigned each PAN the ONS data from their respective LAD, to the detriment of intra-LAD variance. This is an unfortunate workaround but the single one until studies with transactional-to-medical data linkage studies appear.

2.6.3 Unsupervised learning theory

An explanation of PCA is available in section A.4. It is recommended to read it to better understand how t-SNE and UMAP work.

t-SNE

t-SNE is a visualisation method first presented by Hinton and Van Der Maaten in 2008 [51]. t-SNE's goal is to find a 2 or 3 dimensional projection of a higher dimensional data for visualisation. It aims at maintaining the distance between neighbouring points in higher dimensional space in the lower dimensional space. It does this by first calculating similarity scores between points in the high and low dimensional settings, P and Q respectively. Next it minimises the Kullback-Leibler divergence between the distribution of the similarity scores in the high dimensional space and the lower dimensional space using gradient descent:

$$\sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{ij} \cdot log(\frac{p_{ij}}{q_{ij}}) \tag{2.3}$$

where P_i represents the conditional probability distribution over all other data points given data point x_i. p_{ij} and q_{ij} are defined as:

$$\begin{aligned} p_{ij} &= \frac{p_{j|i} + p_{i|j}}{2n} \\ q_{ij} &= \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}} \end{aligned} \tag{2.4}$$

where $p_{b|a}$ represents the conditional probability that the point attributed to x_b under Gaussian curve centred at x_a. Note that q_{ij} correspond to a joint probability that of variables y_i and y_j that follow a Student's t-distribution (with degree of freedom equal to 1).

$$p_{b|a} = \frac{exp(-||x_a - x_b||^2/2\sigma_a^2)}{\sum_{c \neq a} exp(-||x_a - x_c||^2/2\sigma_a^2)} \tag{2.5}$$

σ_a is the variance of the Gaussian centred at x_a.

Two known caveats of t-SNE are that it does not scale well for big datasets and that it does not penalise assigning a short distance to points that are very far away in the original space [51], hence often failing at capturing global distances.

UMAP

UMAP is a general-purpose dimensionality reduction techniques built around a strong theoretical framework of topological data analysis [53]. It is also more computationally tractable which enables its use for visualisation of very high dimensional data in reasonable time-frames for cases such as image data, RNA sequencing data and others [53][56]. While the mathematics behind the theoretical framework of UMAP are advanced, conceptually what UMAP does is simple. UMAP's objective is to preserve the topology of the manifold formed by the data in high dimensional space in a lower dimensional space. In practice, the UMAP algorithm builds a weighted graph where each node is a data point and the weight of the edges is proportional to how close points in the higher dimensional space are. To build this graph, UMAP uses a local notion of distance based on the nearest neighbours algorithm. The number of neighbours can be set as a parameter, where a lower value is better at preserving the local characteristics of the data and greater values are better at preserving the global characteristics of the data [57][56]. The choice of hyper-parameters is still important hence in our analysis we always tested several hyper-parameters to make sure our projection were not spurious.

2.7 Extra: Supplementary work

The results for the supplementary work are available in the appendix B but will not be discussed.

2.7.1 Univariate analysis

We performed univariate analysis of each of the variables selected using linear models. We adjusted each model to total population, age, and female population. We obtained volcano plot showing the effect size and p-value of each of the corresponding variable coefficient with respect to life expectancy. We also report statistical significance as per the Bonferroni correction to adjust for multiple testing:

$$\alpha_{Bonferroni} = \frac{\alpha}{n}$$

where $\alpha_{Bonferroni}$ is the adjusted significance level, α is the original significance level (taken here as the standard 0.05) and n is the number of statistical tests performed (here 1xx).

2.7.2 PCA and correlation analysis

We investigated the PCA projections of the merged Visa and ONS data and label the projected labels with their corresponding UK region.

2.8 Extra: Data Visualisation dashboard

We built a Shiny application to visualise the data. Due to confidentiality we can only share a version of the app that allows visualisation of the ONS data, that you can find at this link. The code to make the dashboard is available in GitHub.

Chapter 3

Results

3.1 Data characterisation

3.1.1 Summary statistics

PAN table & LAD table

The summary table for PAN data is available in 3.1 and for LAD data is available in 3.2.

Outcome variable distributions

The distribution of the 6 main outcome variables (non-scaled) for the model performance comparison experiment are available in figure 3.1. The mean, median and standard deviation of each outcome is also available in the plots for reference.

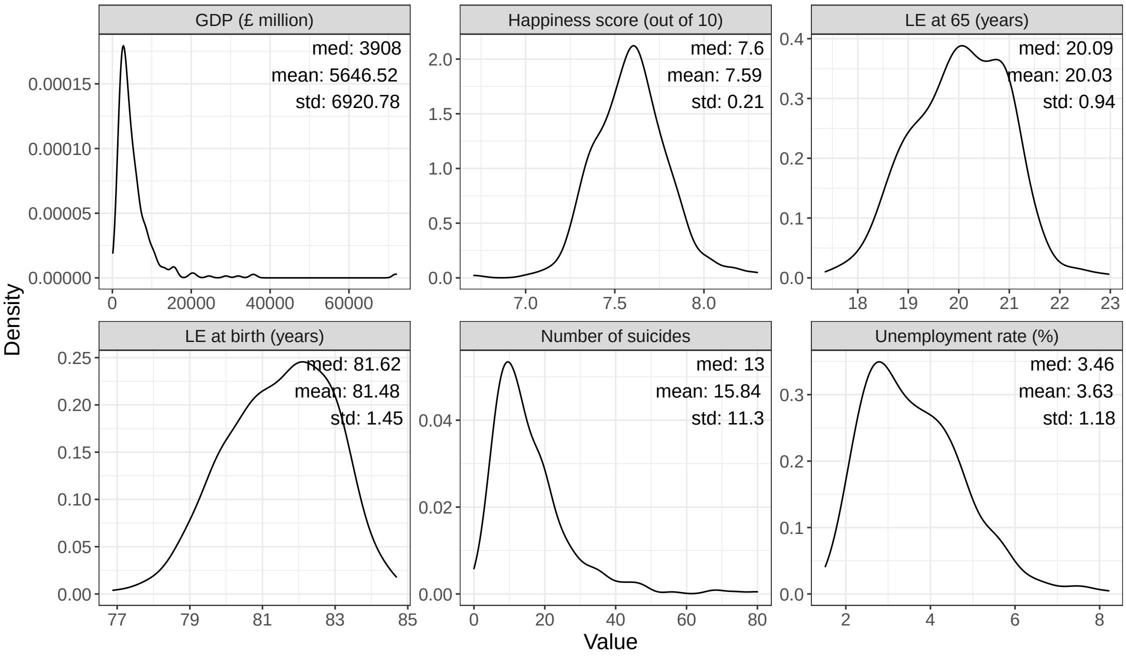


Figure 3.1: Distribution and summary statistics of the main outcome variables

	England (N=22,426,402)	**Wales (N=1,140,283)**	**Overall (N=23,566,685)**
Count of attempted tx (Hundred units)			
Mean (SD)	3.74 (2.64)	3.65 (2.42)	3.73 (2.63)
Median [Min, Max]	3.14 [0.500, 197]	3.18 [0.500, 214]	3.14 [0.500, 214]
Sum spent (Hundred GBP)			
Mean (SD)	107 (138)	100 (96.1)	107 (136)
Median [Min, Max]	81.7 [0.0458, 198000]	80.4 [0.613, 11800]	81.6 [0.0458, 198000]
Average spent per tx (GBP)			
Mean (SD)	36.1 (65.5)	33.7 (47.0)	36.0 (64.7)
Median [Min, Max]	24.8 [0.0101, 39000]	24.5 [0.187, 9870]	24.8 [0.0101, 39000]
St.dev in tx value (GBP)			
Mean (SD)	111 (283)	99.9 (233)	110 (281)
Median [Min, Max]	46.4 [0, 44000]	43.3 [0, 25700]	46.3 [0, 44000]
Count of mobile payments tx (units)			
Mean (SD)	32.6 (109)	25.2 (90.0)	32.3 (108)
Median [Min, Max]	0 [0, 4110]	0 [0, 2180]	0 [0, 4110]
Count of contactless tx (Hundred units)			
Mean (SD)	1.86 (1.91)	1.75 (1.76)	1.85 (1.90)
Median [Min, Max]	1.33 [0, 41.0]	1.29 [0, 23.5]	1.33 [0, 41.0]
Count of card present tx (Hundred units)			
Mean (SD)	2.65 (2.14)	2.60 (1.99)	2.65 (2.13)
Median [Min, Max]	2.16 [0, 42.5]	2.20 [0, 25.3]	2.17 [0, 42.5]
Count of approved tx (Hundred units)			
Mean (SD)	3.69 (2.61)	3.60 (2.38)	3.68 (2.60)
Median [Min, Max]	3.09 [0.500, 113]	3.13 [0.500, 47.6]	3.10 [0.500, 113]
Count of declined tx (units)			
Mean (SD)	4.71 (26.4)	4.92 (33.2)	4.72 (26.8)
Median [Min, Max]	0 [0, 19400]	0 [0, 20600]	0 [0, 20600]
Funding source - Count(% of total)			
Missing	73 (0.0%)	1 (0.0%)	74 (0.0%)
Credit	2362445 (10.5%)	93403 (8.2%)	2455848 (10.4%)
Debit	20058937 (89.4%)	1046716 (91.8%)	21105653 (89.6%)
Charged cards	2656 (0.0%)	94 (0.0%)	2750 (0.0%)
Pre-paid	459 (0.0%)	15 (0.0%)	474 (0.0%)
Deferred debit	1832 (0.0%)	54 (0.0%)	1886 (0.0%)
Number of pans per country - count (% of total)			
England	22426402 (100%)	0 (0%)	22426402 (95.2%)
Wales	0 (0%)	1140283 (100%)	1140283 (4.8%)

Table 3.1: Summary statistics for some PAN data features. *tx stands for transaction, SD for standard deviation*

	England **(N=317)**	**Wales** **(N=22)**	**Overall** **(N=339)**
Count of attempted tx (Million units)			
Mean (SD)	43.6 (31.2)	30.6 (17.7)	42.7 (30.7)
Median [Min, Max]	32.7 [0.384, 242]	26.4 [12.2, 88.3]	31.9 [0.384, 242]
Count of mobile payments tx (Million units)			
Mean (SD)	3.83 (3.45)	2.16 (1.78)	3.72 (3.39)
Median [Min, Max]	2.57 [0.0173, 21.6]	1.69 [0.716, 8.86]	2.47 [0.0173, 21.6]
Count of contactless tx (Million units)			
Mean (SD)	21.4 (15.8)	14.4 (9.10)	20.9 (15.6)
Median [Min, Max]	15.7 [0.156, 112]	11.8 [5.58, 45.4]	15.2 [0.156, 112]
Count of card present tx (Million units)			
Mean (SD)	30.4 (21.6)	21.3 (12.6)	29.8 (21.2)
Median [Min, Max]	23.1 [0.234, 161]	18.2 [8.19, 62.8]	22.4 [0.234, 161]
Count of approved tx (Million units)			
Mean (SD)	42.8 (30.6)	30.0 (17.3)	41.9 (30.0)
Median [Min, Max]	32.3 [0.382, 237]	25.8 [11.9, 86.7]	31.4 [0.382, 237]
Count of declined tx (Thousand units)			
Mean (SD)	806 (689)	605 (340)	793 (673)
Median [Min, Max]	538 [2.44, 5550]	500 [207, 1650]	536 [2.44, 5550]
Sum spent (Million GBP)			
Mean (SD)	1270 (852)	841 (447)	1240 (838)
Median [Min, Max]	982 [13.1, 6400]	765 [325, 2220]	972 [13.1, 6400]
Average spent per tx (GBP)			
Mean (SD)	30.0 (3.28)	27.9 (1.87)	29.8 (3.24)
Median [Min, Max]	29.5 [23.9, 53.8]	27.4 [25.2, 31.9]	29.5 [23.9, 53.8]
St.dev in tx value (GBP)			
Mean (SD)	255 (54.6)	210 (26.7)	252 (54.3)
Median [Min, Max]	247 [148, 561]	209 [161, 271]	244 [148, 561]
Number of cards (Thousand units)			
Mean (SD)	208 (154)	143 (79.6)	204 (151)
Median [Min, Max]	156 [2.12, 1340]	128 [59.4, 379]	151 [2.12, 1340]
Average spent per PAN (GBP)			
Mean (SD)	6240 (943)	5810 (312)	6210 (921)
Median [Min, Max]	6160 [4420, 11300]	5850 [5120, 6460]	6100 [4420, 11300]
St.dev of expenditure per PAN (GBP)			
Mean (SD)	10300 (3590)	8090 (948)	10200 (3520)
Median [Min, Max]	9470 [6350, 47800]	7990 [6360, 9940]	9280 [6350, 47800]
Number of LADs (% of total)			
England	317 (100%)	0 (0%)	317 (93.5%)
Wales	0 (0%)	22 (100%)	22 (6.5%)

Table 3.2: Summary statistics for some LAD data features. *tx stands for transaction, SD for standard deviation*

3.2 Predictive performance

As a note, because we scaled all of our variables, an RMSE of 1 corresponds to an average error of 1 standard deviation.

3.2.1 Life expectancy

The experiment results with life expectancy at birth and at 65 as outcomes are available in figures 3.2 and 3.3 respectively. First let's look at the comparison between ONS data only and Visa data only. In either experiments, the error is almost equivalent if we use ONS data or Visa low data. In both experiments the RMSE decreases when we use the Visa mid data with respect to the ONS data only. For LE at birth, the error decreases by 0.15 and the standard error bars do not overlap between the model run with ONS data only and Visa mid only. For LE at 65 the RMSE decreases only 0.05 and the error bars overlap. This observation remain the same when we use Visa high data. We can see that when we incorporate ONS data with Visa low data the error decreases by 0.08 and 0.02 (for LE at birth and LE at 65 respectively) with respect to the baseline level of using ONS data only or Visa low data only. Incorporating other data combinations improves the error minorly, allowing an average RMSE value of 0.45 and 0.5 (for LE at birth and LE at 65 respectively) to be reached when we incorporate ONS data, Visa mid and the top 5 table. Further combinations of data do not improve the average RMSE score any more.

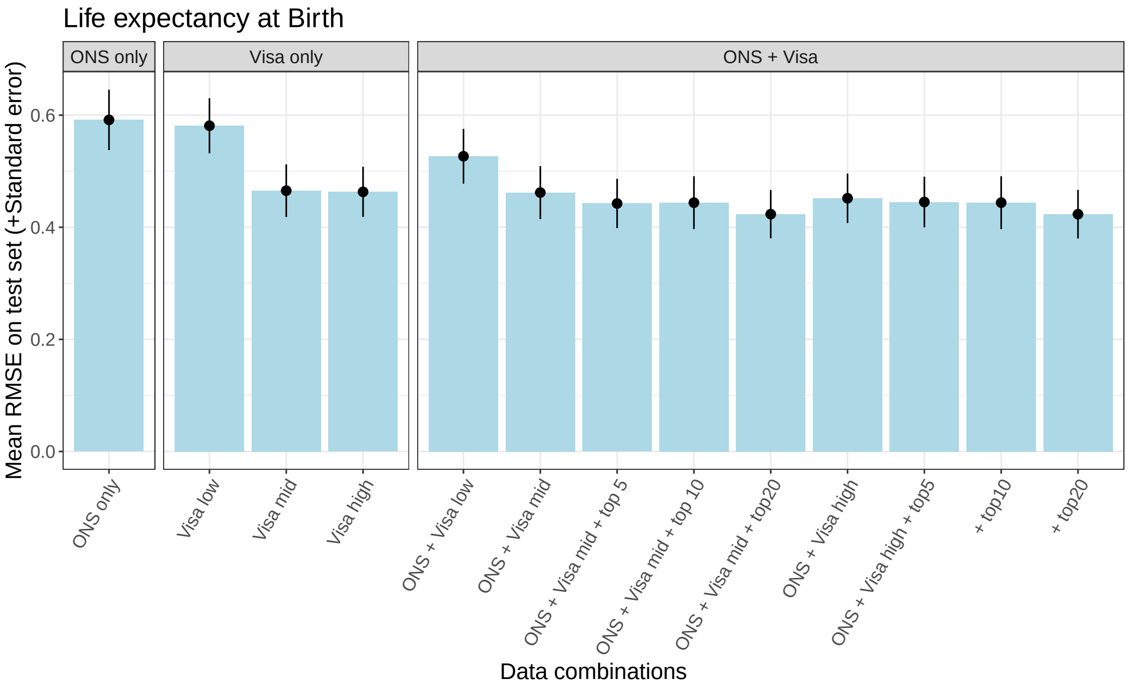


Figure 3.2: Model performance comparison with overall LE at birth as an outcome

3.2.2 Happiness

The experiment results with the average happiness score as an outcome is available in figure 3.4. Here, we obtained the best RMSE for the ONS only model with a value of 0.65. Visa data alone only reached an RMSE of 0.95 which was not improved with greater resolution, i.e. the RMSE was the same for Visa low, Visa mid and Visa high. None of the ONS and Visa combinations improved over the ONS only score.

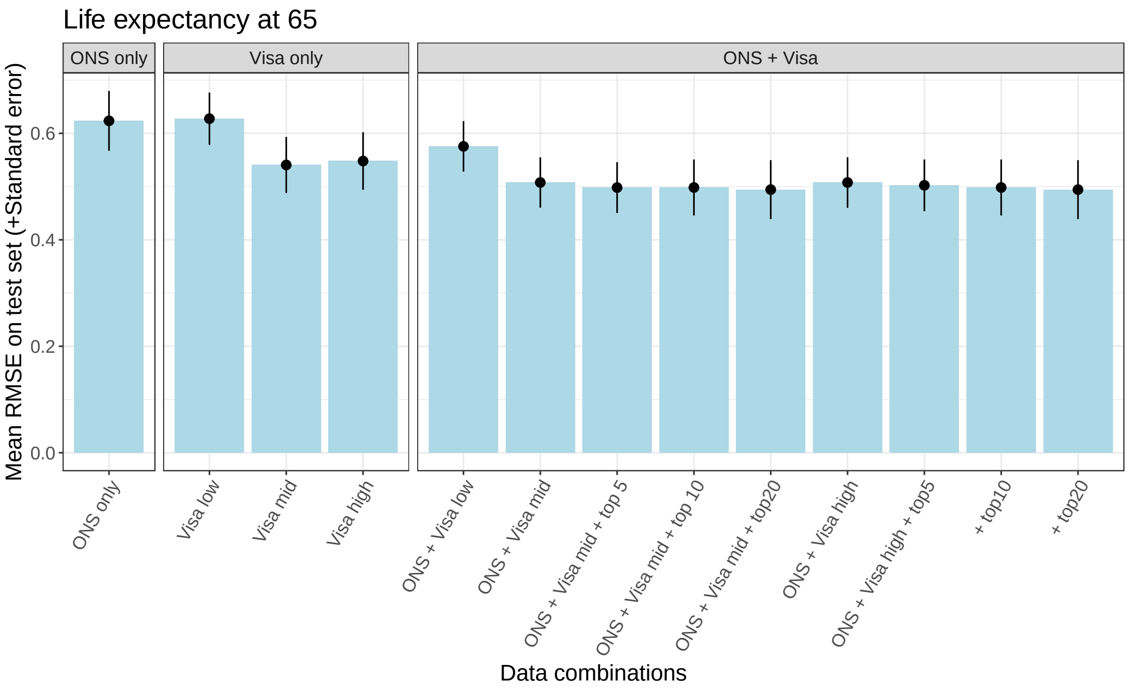


Figure 3.3: Model performance comparison with overall LE at 65 as an outcome

3.2.3 Suicides

The experiment results with the number of suicides as an outcome is available in figure 3.5. In this case, using Visa data alone did not improve over the score from the ONS data only. The RMSE for the model with the ONS data only was of 0.48, while with Visa low the RMSE was of 0.61, with Visa mid and Visa high of 0.52. None of the ONS and Visa combinations improved over the ONS only source.

3.2.4 Unemployment rate

The experiment results with unemployment rate as an outcome is available in figure 3.6. The models using ONS only data achieved an average RMSE of 0.76. In this case, Visa data alone resulted in a reduction of the RMSE. The Visa low models achieved an RMSE of 0.73, where as Visa mid and Visa high one achieved average RMSEs of 0.59. The combinations of ONS and Visa low data did not perform any better than Visa low alone. In a similar fashion no combination of ONS and Visa mid or Visa high data performed any better than Visa mid alone.

3.2.5 GDP

The experiment results with GDP as an outcome is available in figure 3.7. In this case, we observe much larger standard errors ranging from 0.1(with Visa high) to 0.25(with ONS only data) where as in other experiments the largest SE was close to 0.1. Predicting GDP with ONS data only resulted in the highest (worst) RMSE value of 0.76. Using Visa data alone the results improved. Using Visa low data the RMSE obtained was 0.67, with Visa mid data of 0.6 and with Visa high data of 0.37. Combinations of ONS data with Visa mid data did not results in RMSE scores any lower than with Visa mid alone. Combinations of ONS data with Visa high data did not result in RMSE values lowers than with Visa high alone either. The incorporation of top 10 and top 20 data increased the RMSE over the use of ONS, Visa high and top 5 combined.

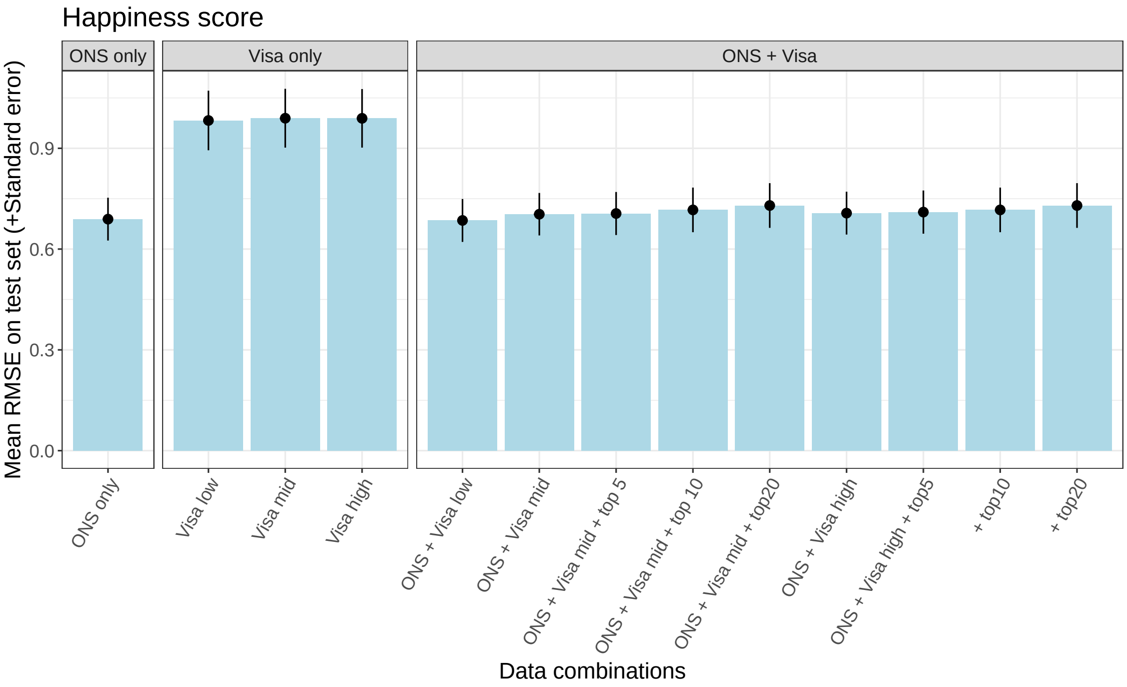


Figure 3.4: Model performance comparison with the average happiness score as an outcome

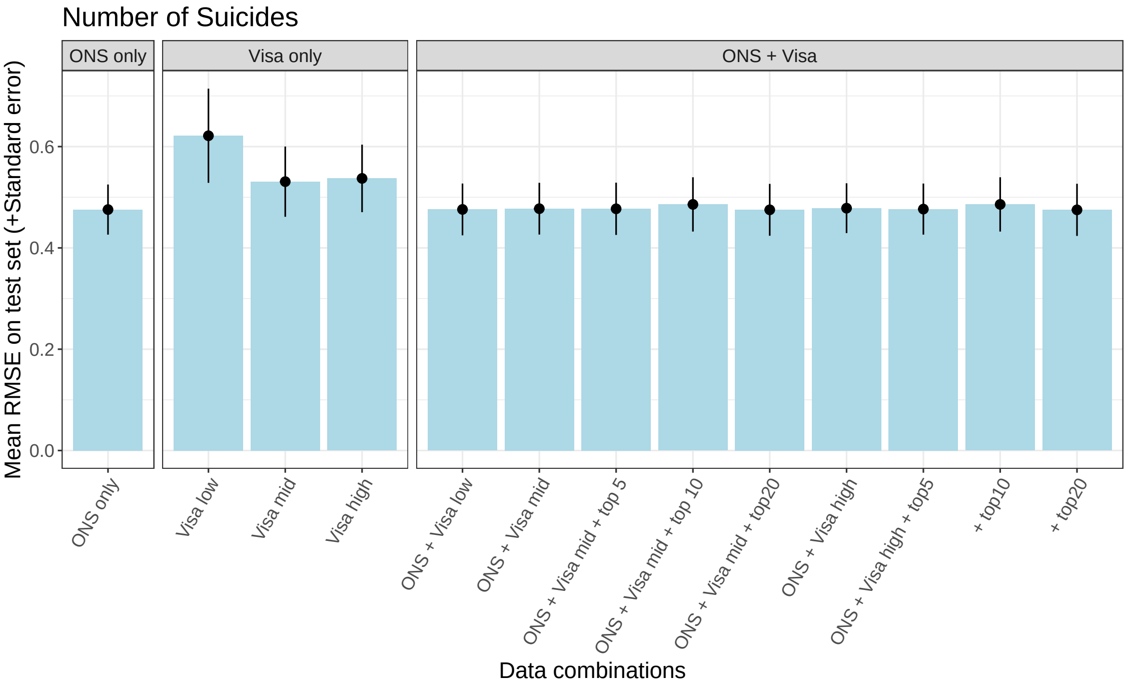


Figure 3.5: Model performance comparison with number of suicides as an outcome

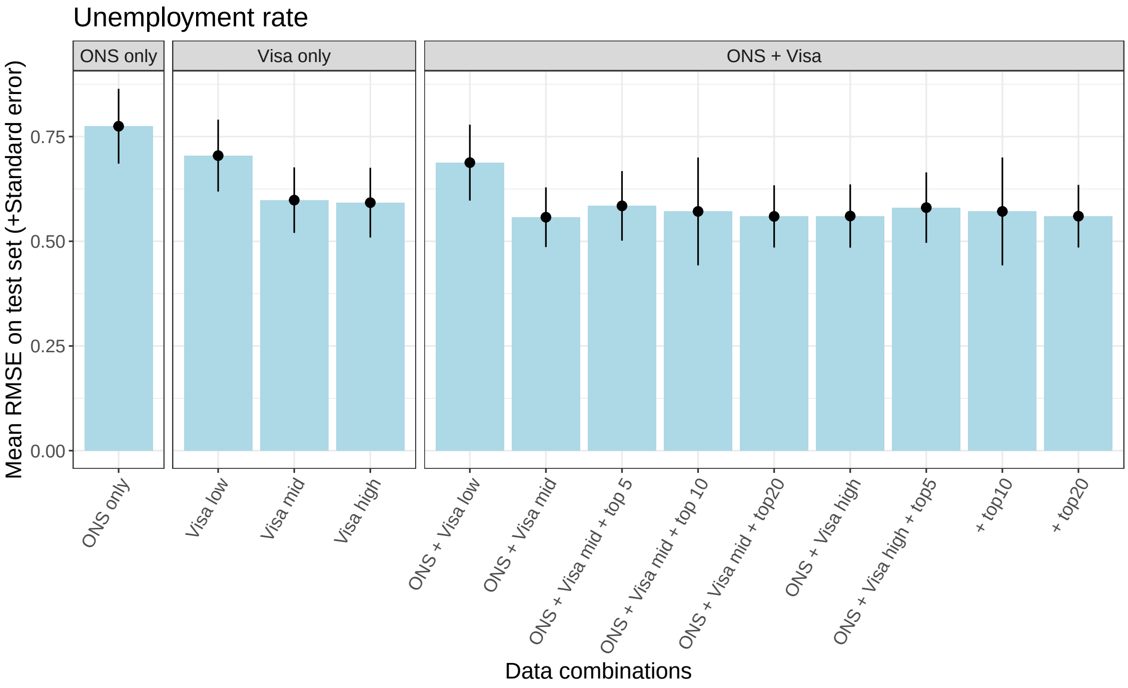


Figure 3.6: Model performance comparison with unemployment rate as an outcome

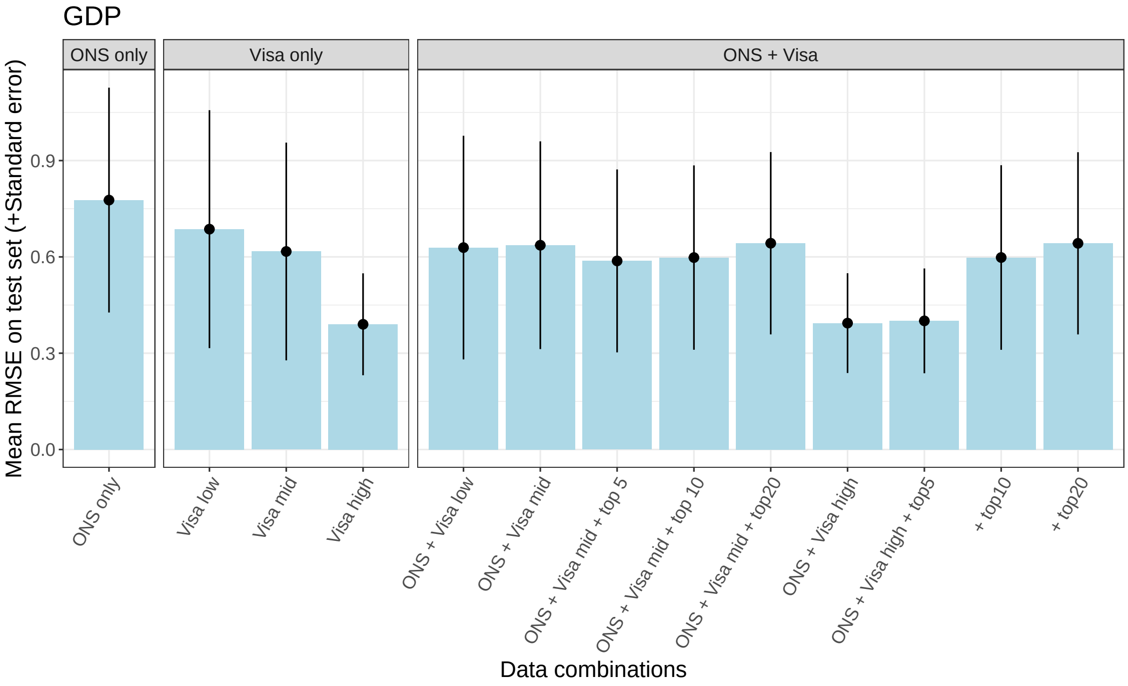


Figure 3.7: Model performance comparison with GDP as an outcome

3.3 Clustering performance

We explored UMAP 2D projections for the LAD and PAN data to evaluate whether the Visa data alone encoded certain external variables. The scale of the plots has been omitted because the 2D coordinate plane the data is embedded to has not interpretable meaning.

3.3.1 LADs

We obtained 2D projections with UMAP with the following hyper-parameter values: number of neighbours equal to 20 and minimum distance in embedded space of 0.1 (relative scale). We labelled the projected samples with external values (not used for the embedding). The projections labelled with LE at birth, unemployment rate, GDP and average happiness score can be found in 3.8 and those labelled by or country can be found in figure 3.9.

In figure 3.8, we observe gradients of the external label values for the LE at birth plot, the unemployment rate and the GDP plots. No clear gradient can be observed in the happiness score plot (bottom right). The observed gradients do not go in the same direction. The LE at birth gradient goes from the top right to the bottom left corner with entries in the bottom left corner corresponding to LADs with greater life expectancy. In the unemployment rate plots, the gradient follows the opposite direction that the LE at birth plot. The unemployment rate value grows from the bottom left to the top right corner of the plot. Finally, in the GDP plot we observe a gradient from the bottom right to the top left corner with points in the top left corner corresponding to those with the greatest GDP (note this gradient is in log scale, the gradient is much steeper in natural scale).

In figure 3.9, we observe the regions and countries the points belong to. First of all we see there is no clear difference between England and Wales LADs by looking at the embedding on the right hand side picture. One thing to note in the right hand side plot is that no Welsh LAD is observed between the centre line and the left side of the plot. The embedding on the left hand side plot, coloured by the corresponding region of each LAD shows that the points corresponding to the London region form a cluster (near the top left corner). No other cluster can be identified.

3.3.2 PANs

We are presenting the embedding results of 100,000 randomly sampled data points out of the 1 million data points the UMAP embedding is based on. Plot B.1 contains the embedding colour by LE at birth of these points with different hyper-parameter values. First of all the embeddings are difficult to visualise clearly because there are many points and no clear formation of clusters to differentiate all these points. We also notice that sometimes the cluster of points appear very small in the plots because of the presence of extremely rare outliers. We see, as expected that with low choice of number of neighbours, local structure is over-preserved and spurious strings of observations are formed. No gradient stands out in the embedding at either of the hyper-parameter settings.

Additionally, figure 3.10 shows the embedding of 1, 10, 100 thousand and 1 million entries. The hyper-parameters for this plot were 100 neighbours and a spread of 10. Visualising the embeddings allows to see that the lack of gradient or clustering in the embeddings is not an artefact of the scale of the visualisation. In fact no gradient appears when plotting at either of these scales.

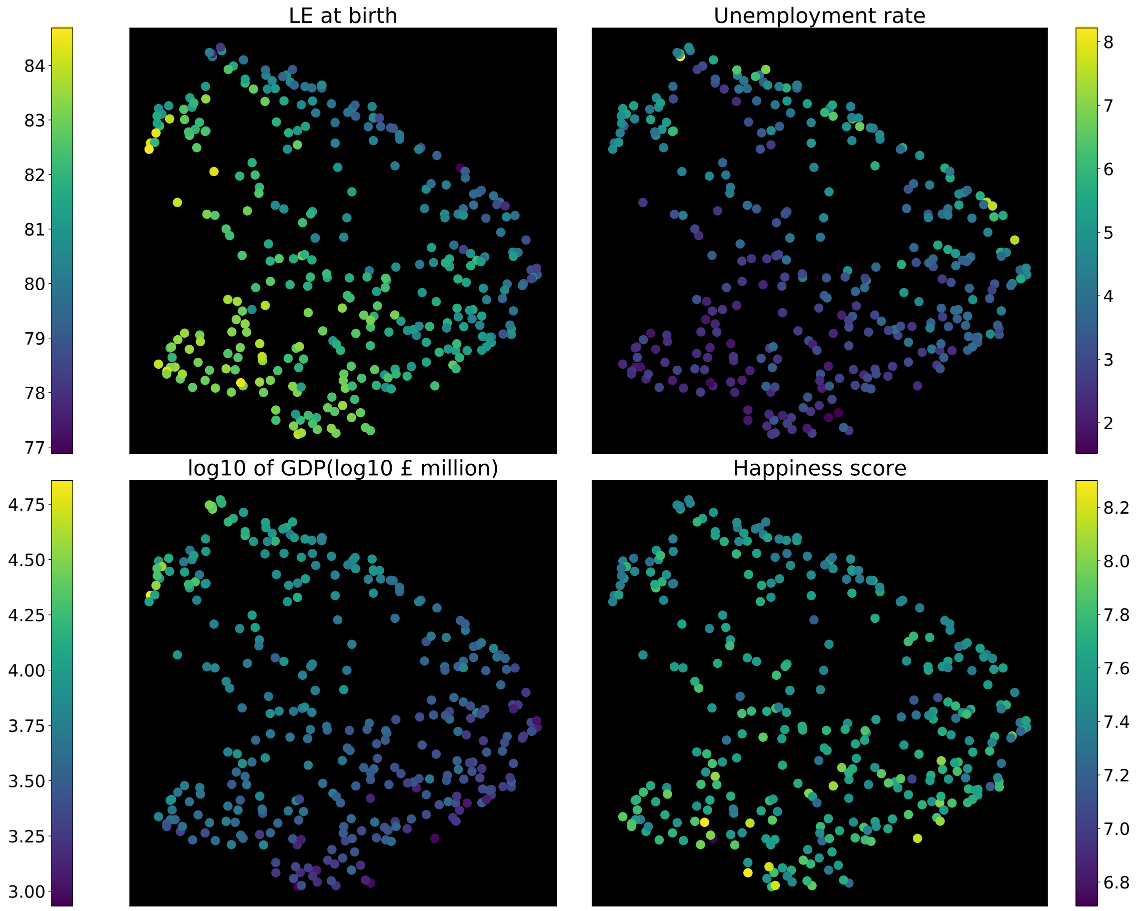


Figure 3.8: UMAP projection of LAD data labelled by LE at birth, unemployment rate, GDP (in log10 of million £) and average happiness score

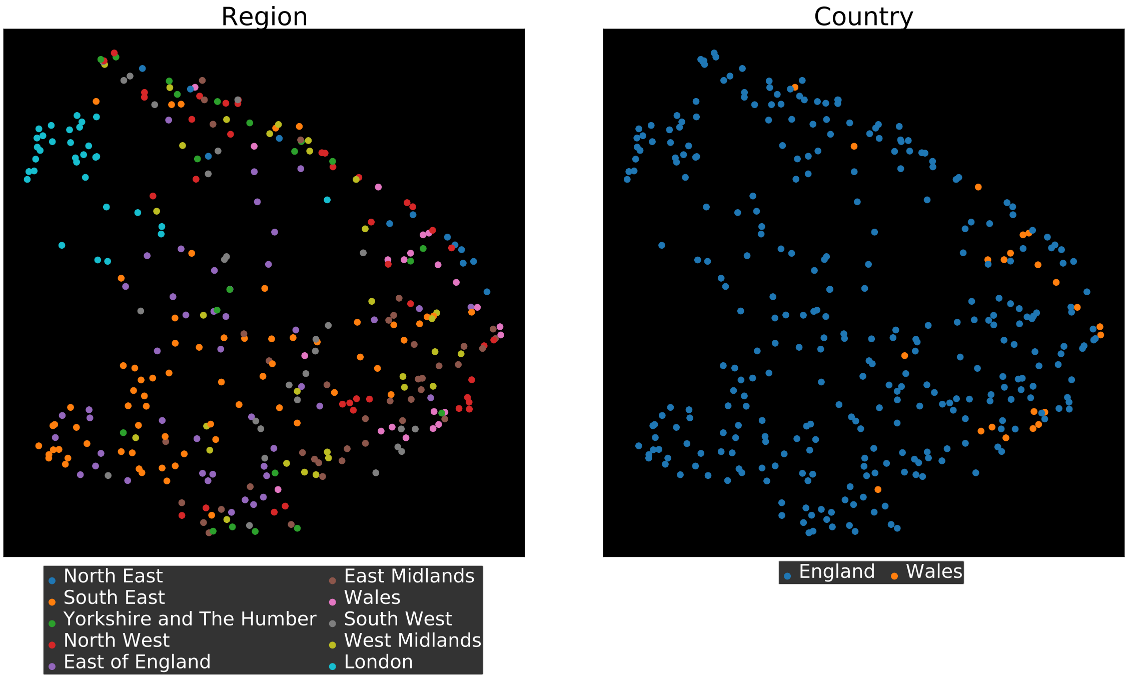


Figure 3.9: UMAP projection of LAD data labelled by region and country

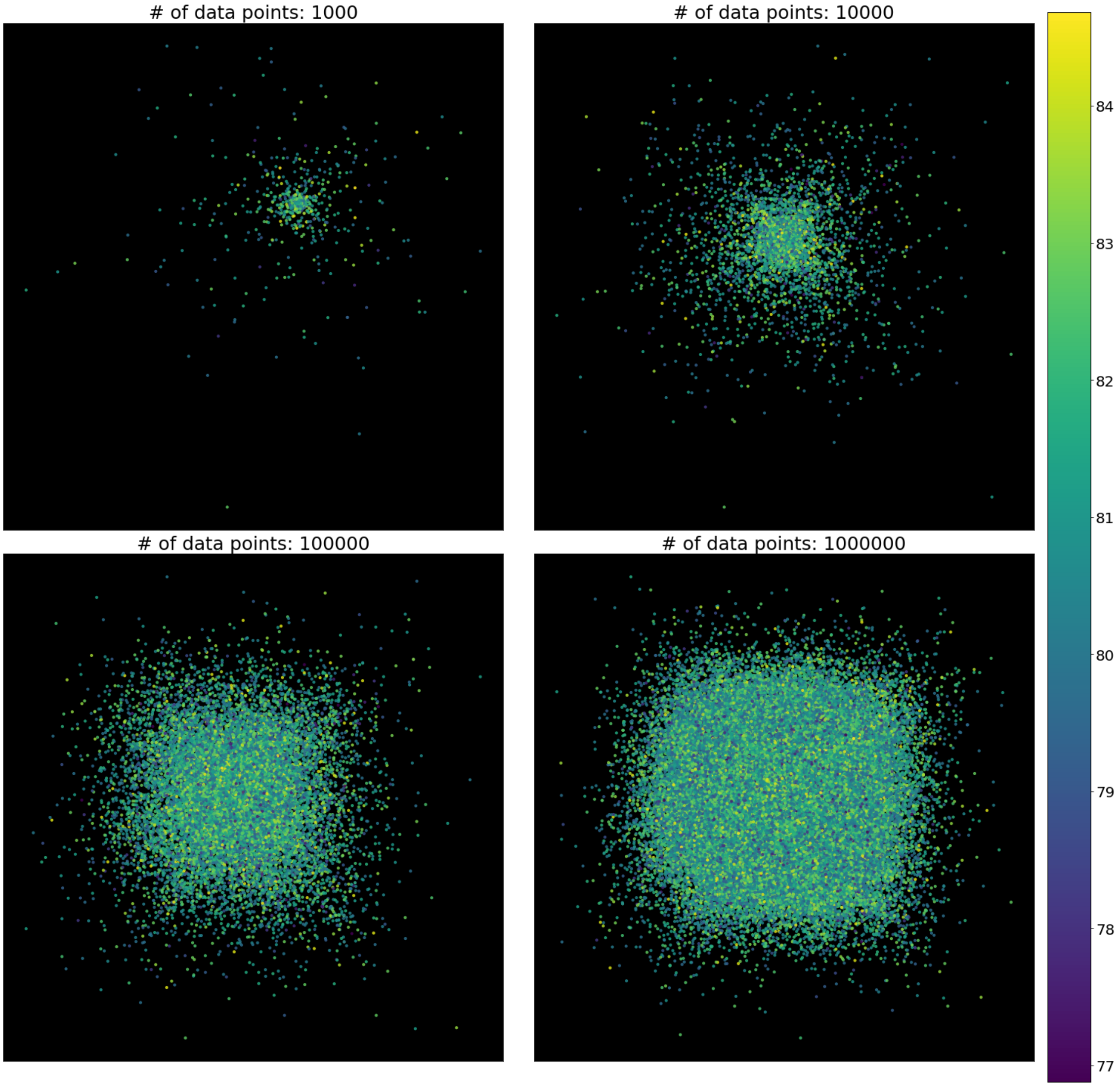


Figure 3.10: UMAP embedding of the PAN data set of different sizes (1000, 10,000, 100,000 and 1,000,000. The dots are coloured by LE at birth

Chapter 4

Discussion

4.1 Transactional data can improve predictive performance on health metrics

The results in this section were mixed. In some instances, the best predictive performance was achieved with Visa data alone, some other times with ONS data alone and in some other cases the combination of both would result in the best result.

For the cases of life expectancy and unemployment rate, the best average result was obtained with a combination of ONS and Visa mid resolution data. Although, the standard error for the model runs with this combination of data overlap with the error bars when using Visa mid resolution data only. It is also worth noting that in the case of unemployment rate, using Visa low resolution which contains only 7 features, the average RMSE score was lower than with the 60 ONS features. It is hard to hypothesise which variables could aid in the prediction of different outcomes, future research should focus on individual outcomes and investigate what Visa or non-Visa features are strong predictors of it.

4.2 Higher behavioural resolution can improved predictive performance on health metrics

We performed two "resolution" experiments. One looking at increased resolution with Visa data alone and the other with increased resolution along with ONS data. To reiterate, the low resolution data included only high level expenditure data such as the number of cards and pan-wise and transaction-wise expenditure statistics. The mid level contained the same information but also information about the count of approved and declined transactions as well as the counts of transactions per mode of payment (contactless, card-present, mobile-payment) and expenditure by MCG. Finally the Visa high data contains extra information regarding the accounts in each LAD, such as the number of accounts per bank.

When testing predictive performance of number of suicides, life expectancy, unemployment rate and GDP increasing the resolution of the Visa data reduced the RMSE. Only in the case of GDP prediction, was the RMSE lower when using Visa high data than when using Visa mid data. In the case of the average happiness score, increasing resolution of Visa data did not result in better performance.

The experiment on predicting GDP suggests that the account-wise information is an excellent predictor of GDP reducing the RMSE by 0.2 standard deviations and narrowing the standard error from nearly 0.4 (for Visa mid data) sd to 0.15 (The GDP standard deviation is equal to £6,920 million). The only difference between Visa mid and Visa high is the account-wise information. Further research is necessary to reveal which variables in Visa high the LASSO model is selecting to make these predictions.

When using even higher resolution data, where the transaction statistics where available by MCG an by enterprise (top 5, top 10 or top 20 ranked enterprises) the result did not improve significantly for any of the predicted variables. Only in the prediction of life expectancy and unemployment did we observed a minor reduction in RMSE when using the "top 20" transaction statistics. We hypothesise this is occurring because the market share of enterprises by MCG at the local authority level maybe homogeneous, i.e. the national ranking of enterprises by MCG is equivalent to the local authority ranking in all or most local authorities. This would need to be verified. Additionally, we are limited to 334 observations (one per LAD) hence the maximum number of variable that LASSO can select is 334 which may be a problem when using very high dimensional data.

These results suggest that increased resolution does improve predictive performance. This confirms the hypothesis that using transactional data at a high resolution level (by MCG) is desired when studying differences in health, social and economical metrics between . Further research is required to investigate the role of account-wise information and of the highest resolution statistics (by MCG by enterprise).

4.3 Successful embedding results with LAD data

We were able to find gradients and even clusters when embedding LAD data into a two-dimensional space. This approach is a highly efficient way of seeing what external information the Visa data carries. Our results suggest that LADs with different life expectancies, unemployment rate and GDP also have different overall transaction patterns. We have also found that London has expenditure patterns different enough to all other LADs, enough for a cluster of London LADs to be formed. With this result we conclude that Visa data at the local authority level encodes health, social and economical information, which encourages exploration of whether this holds when looking at individual transactional patterns and individual health, social or economical metrics.

4.4 Unsuccessful embedding results with PAN data

Unfortunately, embedding the high dimensional PAN data did not result in satisfactory outputs. There is a simple explanation to this. The labels we assigned to each PAN were those of their local authority. Any single local authority is likely to contain a very heterogeneous set of people with corresponding heterogeneous behaviours. The cardholders corresponding to the PANs within any given LAD probably have very different health outcomes within any given LAD but by assigning the health metrics of their local authority we neglect those differences. More over individuals with similar transaction patterns will be plotted nearby in the UMAP embedding but because they may come from different LADs they will be assigned different health outcomes, even though their health outcomes could be related. Additionally, PAN with very dissimilar transaction patterns will not be plotted near each other but if they reside in the same LAD they will be assigned the same label. These artefacts result in a plot that could have been generated by overlapping random 2D Gaussian distributions. We hypothesise that if we had had individual data available we would be successful at finding gradients or cluster in the same way we did with the embedding of LAD data.

It remains to be seen if by comparing PAN from very different LADs (e.g. Westminster and Blackwell, richest and poorest), the PAN information from these LADs results in differential embeddings.

4.5 Limitations

4.5.1 Unavailability of individual health records

The scale of the data at Visa is exceptional, though in the absence of external per PAN or per card-holder information the statistical power the data holds shrinks quickly. In this study we were limited to using health metrics at the local authority which are too large. When considering the per LAD statistics too much information is lost. For reference the mean, median and minimum

LAD population sizes (for 2019) were 174383, 137532 and 2242 (Isles of Scilly) respectively in a total of 339 LADs in England and Wales.

Unfortunately, when looking at higher resolution areas such as Lower-layer Super Output Areas (LSOA), less information is available as the ONS collects less data for smaller areas. Population level data and death records are recorded at the LSOA level [58][59] but life expectancy (for example) is not. For reference, in 2019 in the UKthere were 32,844 LSOAs with an average population size of 1500 [60].

4.5.2 What Visa does not see

Is coverage homogeneous across LADs?

At the time this research took place, statistics about Visa coverage per LAD could not be gathered. If the Visa coverage was not homogeneous across LAD (i.e. constant proportion of coverage with the respect to local transaction volume), re-sampling techniques should be used to adjust for it.

What about cash and the transactions through other systems?

In a similar way, transactions that occur through other media, such as cash or other network payment companies (e.g. MasterCard) could bias our result if not accounted for. Individuals in the UK and elsewhere use several forms of payment, the average number of cards per inhabitant in the UK is of 2.4 [25]. Additionally, our PAN analysis is not equivalent to card holder analysis. Even if individuals had multiple Visa cards, it is not possible to link those cards as being from a single individual. Future research should investigate how to adjust for the unseen transactions and for the multiple forms of payments an individual may use.

4.5.3 Biases in transactional data

A cross-country study by the European Central Bank observed that the preference to use electronic payment technology lowers with age, income and educational level as well as with transaction size (lower value transaction tend to be done with cash) [61]. This other study [62] observed similar findings in the US. It is important to understand these biases as forms of selection bias where the transactional data may only reflect the characteristics of people above a certain level of affluence.

4.6 Future work

4.6.1 Towards cohort studies with transactional data

The greatest limitation in our study was the lack of per PAN or per card-holder information. This greatly reduced the statistical power the Visa data contains. We have proved that at the regional level transactional data alone or transactional data in conjunction with publicly available data can improve the prediction of health related variables. We have also proved that at the regional level transactional data inherently encodes health information. The next step is to prove that our findings are also true at the card-holder level, though for this we require card-holder health information. As mentioned in the introduction early studies suggest that individuals favour the idea of donating their transactional data for public health research [18]. We believe that transactional data is a great tool to measure behaviour and even though it may inherently suffer of selection bias, our view is that in general most clinical studies suffer of similar selection biases, where the poorest individuals are often not studied. Transactional data should be welcomed as a novel tool to measure behaviour and we believe it could improve our understanding of public health as well as the efficacy of public health campaigns.

4.6.2 Different regional or temporal context

It would be interesting to study whether our results hold when applying the same methodology to different regions other than England and Wales or different times, e.g. other years, the present COVID19 pandemic, etc.

McKinsey & Co. have reported the shift in consumer sentiment and behaviour during the pandemic and have observed how consumer intent is still below pre-crisis levels [63]. Additionally, expenditure patterns across different retail categories have changed dramatically [64].

4.6.3 Better ONS data and more outcomes

Despite our efforts to extract as much ONS data, we could be criticised for not collecting enough. We could have collected data regarding deaths per LAD or diseases rates per LAD (diabetes, cancer incidence rate...) but we lacked the time to do so in the project time-frame. Future research should try to incorporate more publicly available data in an effort to make the baseline model (using public data) as competent as possible.

Additionally, we regard as interesting trying to predict the Index for Multiple Deprivation (IMD) per LAD or per LSOA (ideally) which combines income, employment, education, health, crime, housing and environment information to provide an overall deprivation score [60]. The IMD is an expensive and time-consuming score to measure that is only calculated every 3 or 4 years [65]. Continuously monitoring IMD would be a great application of transactional data as this is collected regularly. Finally because IMD is provided as a rank, we esteem rank regression should be ideal model candidate to be used when trying to predict IMD [66].

Chapter 5

Conclusion

In this study, we have proved that transactional data improves prediction of health, social and economical metrics at the local authority level. The concept of behavioural resolution has been proven to be effective, with greater resolution leading to better predictions of health outcomes. We have also shown that transactional data implicitly encodes health related information through the use of dimensional reduction techniques.

We have been limited by the lack of card-holder level health metrics. Future research should focus on integrating such information with transactional data making sure the data privacy concerns of the participants are covered. This will enable to verify whether our findings apply at the individual level and be a step towards the integration of transactional data as a general tool in public health research.

Appendix A

Methods appendix

A.1 Columns in authorisation table

The following information was present for each transaction:

- Acquirer ID
- Amount in EUR and GBP
- AVS postcode
- Card Acceptor ID, equivalent to terminal ID, assigned by the merchants to each available terminal
- Country where the cardholder account is registered
- Default currency of the cardholder's bank account
- Currency of the transaction
- Classification of the type of the transaction: e-commerce, face-to-face, telephone, recurring payments
- Card expiry date
- Funding source
- Global Merchant Repository (GMR) ID, identifier used internally at Visa linking terminals to merchants (i.e. a certain terminal to a certain grocery branch)
- Whether card was present in transaction (face-to-face or not)
- Whether transaction was performed using contactless technology
- Whether a mobile payment system was used (Apple pay, Google pay...)
- Issuer ID
- Merchant Category Code (MCC), code that enables to categorises business into very narrow categories (900) - standard used across network payment companies
- Merchant Category Group (MCG), Visa's own supra-categorisation of MCCs into 28 groups: department stores, fuel, quick service restaurants, professional services, telecom & utilities, food & grocery, business-to-business, transportation, general retail goods, online marketplaces, education & government, direct marketing, general retail services, automotive, healthcare, drugstores & pharmacies, lodging, home improvement & supply, discount stores,vehicle rental, entertainment, restaurants, wholesale clubs, apparel & accessories, electronics, airlines, travel services, insurance.

- Country where the merchant bank account is registered
- Merchant name (as registered by the merchant, usually what appears in a receipt)
- PAN
- Type of Visa card used (Visa debit, Visa electron, Visa Platinum...
- Purchase date, purchase time
- Response code and response code group relating to what happened to the transaction, was it approved, declined? Reason for approval/decline?
- Whether transaction was domestic (cardholder and merchant registered in the same country), intra-regional (as per Visa's definition of regions) or inter-regional.

A.2 ONS columns - Clean dataset

- LAD code and name (2 features)
- Population count of male, females, overall. (3 features)
- Population count of teens (< 17 years old), young adults (between 17 and 30), adults (between 30 and 64), elderly (64 and 81) and very elderly (older than 81) – Stratified by gender and overall (15 features)
- Mean, median, mode and standard deviation of age stratified by gender and overall (12 features)
- Region the LAD belongs to (1 feature)
- Life expectancy estimates at birth and at 65 stratified by gender (4 features)
- Life expectancy confidence interval values (lower confidence interval and higher confidence interval) at birth and at 65 stratified by gender (8 features)
- GDP, unemployment rate (2 features)
- Number of suicides (1 feature)
- Average and range of the scores of the 4 wellbeing metrics (8)
- Mean and median hourly pay for full-time, part-time and overall jobs stratified by gender and overall (18 features)
- Coefficient of variation (taken as an estimate of the difference between the population and sample estimates) of the mean and median hourly pay for full-time, part-time and overall jobs stratified by gender and overall (18 features)

A.3 ONS data links

Links available in data A.1.

A.4 PCA theory

PCA is a dimensionality-reduction technique introduced by Pearson in 1901[67]. The PCA method projects data from an original m-dimensional space by finding latent dimensions of maximum variance. These latent dimensions are simply linear weighted sums of the original components:

$$\mathbf{p}_{i,a} = x_{i,1}\ w_{1,a} + x_{i,2}\ w_{2,a} + \ldots + x_{i,k}\ w_{k,a} + \ldots + x_{i,K}\ w_{K,a} \tag{A.1}$$

Dataset	Links
Life expectancy	https://www.ons.gov.uk/datasets/life-expectancy-local-authority/editions/time-series/versions/1
Suicides 1	https://www.ons.gov.uk/datasets/suicides-in-the-uk/editions/time-series/versions/1
Suicides 2	https://www.ons.gov.uk/datasets/suicides-in-the-uk/editions/2018/versions/1
Wellbeing	https://www.ons.gov.uk/datasets/wellbeing-local-authority/editions/time-series/versions/2
Work	https://www.ons.gov.uk/datasets/ashe-tables-7-and-8/editions/2019/versions/1
GDP	https://www.ons.gov.uk/economy/grossdomesticproductgdp/datasets/regionalgrossdomesticproductlocalauthorities
Population 1	https://www.ons.gov.uk/datasets/mid-year-pop-est/editions/time-series/versions/4
Population 2	https://www.ons.gov.uk/datasets/mid-year-pop-est/editions/mid-2018-april-2019-geography/versions/1
Unemployment	https://www.ons.gov.uk/employmentandlabourmarket/peoplenotinwork/unemployment/datasets/modelledunemploymentforlocalandunitaryauthoritiesm01

Table A.1: ONS data links

where i is the an observation from our data matrix X and a refers to the principal component a; for each PC **p**, **w** is a vector of weights by which each value from the original dimension will be multiplied by [68].

In matrix form, the transformation is:

$$\mathbf{P} = \mathbf{X}\mathbf{W} \tag{A.2}$$

Where X is the data matrix (N entries, P columns), W is the weight matrix (P rows (original dimensionality) and A columns (projected dimensionality)) and PC is the projected data matrix (N rows, A columns

The first PC is hence the a latent dimension across which there is most variance. The second PC is the dimension with most variance that is orthogonal to the 1st PC. Every subsequent component must be orthogonal to all the previous PC. The orthogonality constraint decorrelates the data, i.e. all PCs have a correlation of 0. Furthermore each PC is constrained to have a norm of 1.

Algorithmically, PCA uses eigenvalue decomposition to find the PC components. The optimisation problem in question is as follows. The PCs are found recurrently, the first one is found by finding the weights that maximises the covariance of the projection $p_1^T p_1$:

$$\begin{aligned} \max \phi &= \mathbf{p}_1^T \mathbf{p}_1 \\ \text{such that: } \mathbf{p}_1^T \mathbf{p}_1 &= 1 \\ \text{where } \mathbf{p}_1^T \mathbf{p}_1 &= \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \end{aligned} \tag{A.3}$$

Note that the covariance can be calculated in the way stated before because PC requires that the data is mean centred first.

Using a Lagrange multiplier λ_1 for the optimisation constraints yields the following, which has an analytical solution

$$
\begin{aligned}
\max \phi &= \mathbf{p}_1^T \mathbf{X}^T \mathbf{X} \mathbf{p}_1 - \lambda_1 (\mathbf{p}_1^T \mathbf{p}_1 - 1) \\
\frac{\partial \phi}{\partial \mathbf{p}_1} &= 0 = \mathbf{p}_1^T \mathbf{X}^T \mathbf{X} \mathbf{p}_1 - \lambda_1 (\mathbf{p}_1^T \mathbf{p}_1 - 1) \\
\mathbf{X}^T \mathbf{X} \mathbf{p}_1 &= \lambda_1 \mathbf{p}_1
\end{aligned}
\tag{A.4}
$$

Hence λ_1 is an eigenvalue of $\mathbf{X}^T\mathbf{X}$ and $\mathbf{p}_1$ is the corresponding eigenvector. It can be shown that λ_1 is equal to the variance of the first component [69]. This last feature is often exploited to extract the components that ensure a certain percentage of the original variance is kept.

A.5 Cleaning ONS dataset

Starting in next page (38 pages long), it may be more convenient to visit this link.

ONS LAD 2019 data cleaning

Anonymous

Summer 2020

Data cleaning report

Load libraries

```r
library(tidyverse)
library(lubridate)
library(readxl)
theme_set(theme_bw())

library(imputeTS)# time series imputation
library(naniar)# for missing data visualisation

library(knitr) # for rmarkdown rendering

source('time_impute.R', echo = T) # time series wrapper imputation function
```

```
##
## > time_impute = function(df, ..., cols_to_impute = c("Value",
## +     "lowerCI", "upperCI"), year_column = "year") {
## +     dots = as.character(ensyms(. .... [TRUNCATED]
```

```r
source('updateCodes.R', echo = T)
```

```
##
## > updateCodes = function(df) {
## +     if (any(df$areaName == "Shepway")) {
## +         df[df$areaName == "Shepway", "areaName"] = "Folkestone and Hythe"
##  .... [TRUNCATED]
```

Load Data

```r
# deaths = read.csv('OriginalData/deathrecords2020.csv') # caveat: only 2020
# deaths = read.csv('OriginalData/DEATHS02010_2019.csv') # caveat: only by wider regions (around 13)
# -- not using for now as only 2020 deaths

lifeexpect = read.csv('OriginalData/lifeexpectancies.csv')

# population = read_xlsx('OriginalData/popestimates.xlsx',
#                         sheet = 'Dataset',
#                         range = cell_rows(c(3, 121433)))
population = read.csv('OriginalData/POPULATION2018_2019.csv')
```

```
suicides = read.csv('OriginalData/suicide2018.csv')

wellbeing = read.csv('OriginalData/wellbeing.csv')

work = read.csv('OriginalData/working.csv')
work_ts = read.csv('OriginalData/working_time_series.csv')

gdp = read_xlsx('OriginalData/regionalgrossdomesticproductgdplocalauthorities.xlsx',
                sheet = 'Table 5',
                range = cell_rows(c(2, 384)))

unemployment = read_xls('OriginalData/unemployment.xls',
                        range = 'A3:HA375', sheet = 3)
```

Clean data

Here I take data cleaning on a case-by-case basis: tidying and imputing each data set at a time, indepently of each other. A few comments that apply to all the ONS datasets: all data sets appear to have redundant columns (as in duplicates), secondly for some of them the data corresponds to an interval and for some others the data is available for every year.

Deaths table (not used in the end)

```
# deaths table: some columns are duplicate so we remove those, we also change the week strings to week

# str(deaths)
#
# deaths = deaths %>%
#   select(-c(Data.Marking,calendar.years,week, cause.of.death, place.of.death, registration.or.occurre
#   rename(Deaths = V4_1, year = time, areaCode = admin.geography,
#          areaName = geography, deathCause = causeofdeath,
#          deathPlace = placeofdeath, week = week.number) %>%
#   mutate(week = as.integer(str_replace(week, 'week-', '')))
```

Life expectancy table

```
str(lifeexpect)
```

Visualise raw data

```
## 'data.frame':    15768 obs. of  12 variables:
##  $ V4_3                      : num  78.1 56.1 60.9 65.3 NA ...
##  $ lower.confidence.limit    : chr  "77.44868" "54.45824" "59.15845" "63.54933" ...
##  $ upper.confidence.limit    : chr  "78.65818" "57.69662" "62.65689" "67.0574" ...
##  $ Data_Marking              : chr  "" "" "" "" ...
##  $ two.year.intervals        : chr  "2013-15" "2013-15" "2013-15" "2013-15" ...
##  $ time                      : chr  "2013-15" "2013-15" "2013-15" "2013-15" ...
##  $ admin.geography           : chr  "E06000003" "E06000009" "E08000022" "E06000007" ...
##  $ geography                 : chr  "Redcar and Cleveland" "Blackpool" "North Tyneside" "Warrington" .
##  $ life.expectancy.variable: chr  "life-expectancy" "disability-free-life-expectancy" "healthy-life-
##  $ lifeexpectancyvariable    : chr  "Life expectancy" "Disability-free life expectancy" "Healthy life
##  $ birth.cohort              : chr  "birth-males" "birth-males" "birth-males" "birth-males" ...
##  $ birthcohort               : chr  "Males at birth" "Males at birth" "Males at birth" "Males at birth
```

Modify initial data As there are many duplicated columns, we will delete those that are either duplicated or empty and give better names to the ones we keep, as well as changing the type of numeric variables that are registered as strings

```
lifeexpect = lifeexpect %>%
  select(-c(Data_Marking,
            two.year.intervals,
            life.expectancy.variable,
            birth.cohort)) %>%
  rename(Value = V4_3,
         lowerCI = lower.confidence.limit,
         upperCI = upper.confidence.limit,
         year = time,
         areaCode = admin.geography,
         areaName = geography,
         LEvariable = lifeexpectancyvariable,
         cohort = birthcohort) %>%
  mutate(lowerCI = as.numeric(lowerCI),
         upperCI = as.numeric(upperCI))
```

Modify year intervals to years The life expectancy information is available in the year intervals `r{unique(lifeexpect$year)}`, to be consistent with the datasets that are available for every year we turn intervals into years. We do this by taking the upper value of each interval as the year column. e.g.: turning 2013-15 to 2015 and 2014-16 to 2016 and so forth

```
lifeexpect$year = as.numeric(lapply(strsplit(lifeexpect$year, '-'), '[[',1))+2
```

Check missing data with respect to several factors Now we check missing data in the `Value` column by cohort, type of life expectancy variable and year

```
kable(table(is.na(lifeexpect$Value), lifeexpect$cohort)) # all cohorts with same amount of data
```

	Females at age 65	Females at birth	Males at age 65	Males at birth
FALSE	2505	2505	2505	2505
TRUE	1437	1437	1437	1437

```
kable(table(is.na(lifeexpect$Value), lifeexpect$LEvariable)) # lifeexpectancy with the least missing da
```

	Disability-free life expectancy	Healthy life expectancy	Life expectancy
FALSE	2796	2796	4428
TRUE	2460	2460	828

```
kable(table(is.na(lifeexpect$Value), lifeexpect$year)) # 2017 with most missing data
```

	2015	2016	2017
FALSE	3576	3624	2820
TRUE	1680	1632	2436

```
lifeexpect = lifeexpect %>%
  mutate(LEvariable = ifelse(LEvariable == 'Disability-free life expectancy',
                             'DFLE',
                             ifelse(LEvariable == 'Healthy life expectancy',
                                    'HLE',
                                    'LE')),
         cohort = ifelse(cohort == 'Females at age 65',
                         'FemAt65',
                         ifelse(cohort == 'Males at age 65',
                                'MaleAt65',
                                ifelse(cohort == 'Females at birth',
                                       'FemAtBirth',
                                       'MaleAtBirth'))))
```

Shortening strings that are too long

Check number of entries with missing values for all years Here we wish to know how many entries, defined by unique combinations of the local authority code (`areaCode`), the life expectancy(LE) variable (`LEvariable`, e.g. healthy LE, Disability-free LE…) and cohort (`cohort`, e.g. Male at birth, Female at 65…) have no available values (aka all values missing) for all available years.

```
count = 0
combs = 0
for (reg in unique(lifeexpect$areaCode))
  for (var in unique(lifeexpect$LEvariable))
    for(coh in unique(lifeexpect$cohort))
    {
      combs = combs +1

      if (all(is.na(lifeexpect %>% filter(areaCode == reg, LEvariable == var, cohort == coh) %>% select
        count = count+1
      }
    }

cat(paste(count, 'combinations out of', combs, 'combinations have all missing entries'))
```

```
## 1632 combinations out of 5256 combinations have all missing entries
```

```
lifeexpect %>%
  group_by(areaCode,
           LEvariable,
           cohort) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value)),
            ProprortionOfMissingEntries = paste0(round(mean(is.na(Value)),3)*100,' %')) %>%
  group_by(NumberOfMissingEntries,
           ProprortionOfMissingEntries) %>%
  summarise(Count = n()) %>%
  kable()
```

Missingness count by number of missing year entries

NumberOfMissingEntries	ProprortionOfMissingEntries	Count
0	0 %	2772
1	33.3 %	852
3	100 %	1632

Time series imputation First we order the data by year, the time series works assuming the data is ordered.

```
lifeexpect = lifeexpect %>% arrange(areaCode, LEvariable, cohort, year)

lifeexpect = time_impute(lifeexpect, areaCode, LEvariable, cohort,
                         year_column = 'year')
```

```
## [1] "Missingness before imputation (%):"
##     Value   lowerCI   upperCI
## 0.3645358 0.3645358 0.3645358
## Time difference of 1.197485 mins
## [1] "Missingness after imputation (%):"
##     Value   lowerCI   upperCI
## 0.3105023 0.3645358 0.3645358
```

```
lifeexpect %>%
  group_by(areaCode,
           LEvariable,
           cohort) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value)),
            ProportionOfMissingEntries = mean(is.na(Value))) %>%
  group_by(NumberOfMissingEntries,
           ProportionOfMissingEntries) %>%
  summarise(Count = n()) %>%
  kable()
```

Final check on missingness count

NumberOfMissingEntries	ProportionOfMissingEntries	Count
0	0	3624
3	1	1632

Gather data in its final form We finally filter for the latest year (in this case 2017) and pivot the data to get a tidy format. We also keep a copy of the tidy time-series data (just in case)

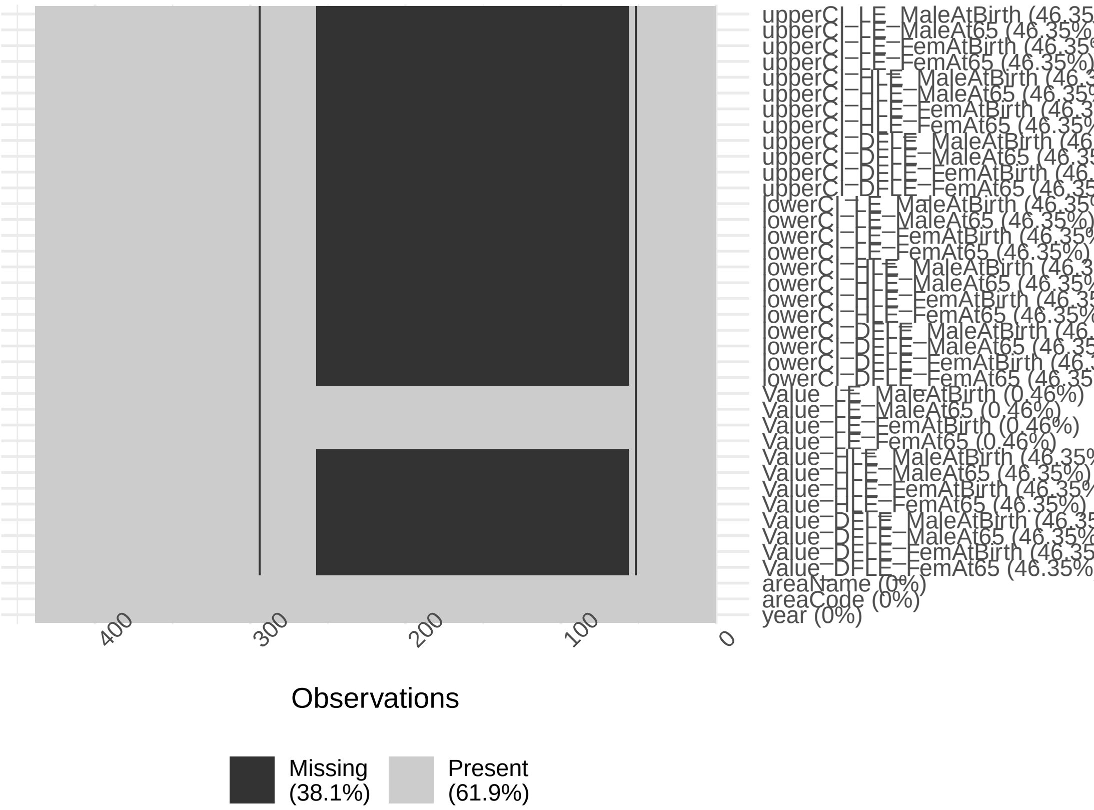
```
lifeexpect_time = lifeexpect %>%
  pivot_wider(names_from = c(LEvariable, cohort),
              values_from = c(Value, lowerCI, upperCI))

lifeexpect = lifeexpect_time %>%
  filter(year == 2017)

str(lifeexpect)
```

```
## tibble [438 x 39] (S3: tbl_df/tbl/data.frame)
##  $ year                    : num [1:438] 2017 2017 2017 2017 2017 ...
##  $ areaCode                : chr [1:438] "E06000001" "E06000002" "E06000003" "E06000004" ...
##  $ areaName                : chr [1:438] "Hartlepool" "Middlesbrough" "Redcar and Cleveland" "Stockt
##  $ Value_DFLE_FemAt65      : num [1:438] 7.42 7.45 9.28 9.63 10.04 ...
##  $ Value_DFLE_FemAtBirth   : num [1:438] 56.2 58 61.2 61.5 60.9 ...
##  $ Value_DFLE_MaleAt65     : num [1:438] 7.56 8.13 9.33 7.58 9.44 ...
##  $ Value_DFLE_MaleAtBirth  : num [1:438] 55.1 57.6 59.3 57.3 60.9 ...
##  $ Value_HLE_FemAt65       : num [1:438] 8.22 8.31 11 10.26 10.47 ...
##  $ Value_HLE_FemAtBirth    : num [1:438] 57 57.6 61.4 60.9 64.3 ...
##  $ Value_HLE_MaleAt65      : num [1:438] 7.67 8.82 9.89 8.15 9.21 ...
##  $ Value_HLE_MaleAtBirth   : num [1:438] 57.4 58.1 58.8 56.6 60.7 ...
##  $ Value_LE_FemAt65        : num [1:438] 19.9 18.9 20.3 19.9 20.3 ...
##  $ Value_LE_FemAtBirth     : num [1:438] 81.4 79.9 81.5 81.4 82.3 ...
##  $ Value_LE_MaleAt65       : num [1:438] 17.1 16.5 18 18.1 18.2 ...
##  $ Value_LE_MaleAtBirth    : num [1:438] 76.1 75.7 77.7 78.1 78.4 ...
##  $ lowerCI_DFLE_FemAt65    : num [1:438] 6.22 6.14 7.9 8.08 8.61 ...
##  $ lowerCI_DFLE_FemAtBirth : num [1:438] 54.3 56.1 59.4 59.5 59 ...
##  $ lowerCI_DFLE_MaleAt65   : num [1:438] 6.39 6.71 8.14 5.94 8.02 ...
##  $ lowerCI_DFLE_MaleAtBirth: num [1:438] 53.1 55.7 57.3 55.1 59 ...
##  $ lowerCI_HLE_FemAt65     : num [1:438] 7.1 7.14 9.71 8.77 9.04 ...
##  $ lowerCI_HLE_FemAtBirth  : num [1:438] 55.3 55.8 59.6 58.8 62.4 ...
##  $ lowerCI_HLE_MaleAt65    : num [1:438] 6.64 7.46 8.81 6.49 7.9 ...
##  $ lowerCI_HLE_MaleAtBirth : num [1:438] 55.8 56.2 56.8 54.4 58.9 ...
##  $ lowerCI_LE_FemAt65      : num [1:438] 19.4 18.5 19.9 19.6 19.9 ...
##  $ lowerCI_LE_FemAtBirth   : num [1:438] 80.7 79.3 81 80.9 81.7 ...
##  $ lowerCI_LE_MaleAt65     : num [1:438] 16.7 16.1 17.6 17.8 17.8 ...
##  $ lowerCI_LE_MaleAtBirth  : num [1:438] 75.3 75.1 77.1 77.6 77.7 ...
##  $ upperCI_DFLE_FemAt65    : num [1:438] 8.62 8.76 10.66 11.18 11.47 ...
##  $ upperCI_DFLE_FemAtBirth : num [1:438] 58 59.8 63.1 63.5 62.9 ...
##  $ upperCI_DFLE_MaleAt65   : num [1:438] 8.73 9.56 10.52 9.22 10.85 ...
##  $ upperCI_DFLE_MaleAtBirth: num [1:438] 57 59.5 61.2 59.5 62.8 ...
##  $ upperCI_HLE_FemAt65     : num [1:438] 9.35 9.48 12.28 11.75 11.9 ...
##  $ upperCI_HLE_FemAtBirth  : num [1:438] 58.8 59.4 63.3 62.9 66.1 ...
##  $ upperCI_HLE_MaleAt65    : num [1:438] 8.7 10.18 10.98 9.81 10.52 ...
##  $ upperCI_HLE_MaleAtBirth : num [1:438] 59 59.9 60.7 58.9 62.5 ...
##  $ upperCI_LE_FemAt65      : num [1:438] 20.3 19.3 20.6 20.3 20.7 ...
##  $ upperCI_LE_FemAtBirth   : num [1:438] 82 80.4 82.1 81.8 82.9 ...
##  $ upperCI_LE_MaleAt65     : num [1:438] 17.6 16.9 18.3 18.4 18.7 ...
##  $ upperCI_LE_MaleAtBirth  : num [1:438] 76.8 76.3 78.3 78.6 79.1 ...
```

```r
vis_miss(lifeexpect) + coord_flip()
```



Visualise missing data

We notice the variables DFLE and HLE have a lot of missingness hence we only keep the LE variables

```r
lifeexpect_time = lifeexpect_time %>% select(year, areaCode, areaName, contains('_LE_'))
lifeexpect = lifeexpect %>% select(year, areaCode, areaName, contains('_LE_'))
```

Update old area codes Pass through custom function to update old codes, then combine the entries with matching area name and area code

```r

lifeexpect = updateCodes(lifeexpect)

combs_dup = lifeexpect %>%
  group_by(areaCode, areaName, year) %>% summarise(duplicat = n()>1)

print(
  lifeexpect[lifeexpect$areaCode %in%
               combs_dup$areaCode[combs_dup$duplicat==T],c(2,3,6,7)])
```

For lifeexpect

```
## # A tibble: 6 x 4
##   areaCode  areaName                  Value_LE_MaleAt65 Value_LE_MaleAtBirth
##   <chr>     <chr>                                 <dbl>                <dbl>
## 1 E07000246 Somerset West and Taunton              18.9                 79.4
## 2 E07000246 Somerset West and Taunton              20.0                 81.0
## 3 E07000245 West Suffolk                           19.5                 80.6
## 4 E07000245 West Suffolk                           20.3                 81.6
## 5 E07000244 East Suffolk                           20.2                 81.8
## 6 E07000244 East Suffolk                           19.1                 79.0
```

This function combines entries by taking the mean when they are duplicated

```
for (row in which(combs_dup$duplicat==T)){
  entry = combs_dup[row,]

  lifeexpect[(lifeexpect$areaCode == entry$areaCode &
                lifeexpect$areaName == entry$areaName &
                lifeexpect$year == entry$year),
             !colnames(lifeexpect) %in% c('areaCode','areaName', 'year')] =
    t(apply(lifeexpect[(lifeexpect$areaCode == entry$areaCode &
                lifeexpect$areaName == entry$areaName &
                lifeexpect$year == entry$year),
             !colnames(lifeexpect) %in% c('areaCode','areaName', 'year')],2,mean))

}

print(
  lifeexpect[lifeexpect$areaCode %in%
               combs_dup$areaCode[combs_dup$duplicat==T],c(2,3,6,7)])
```

```
## # A tibble: 6 x 4
##   areaCode  areaName                  Value_LE_MaleAt65 Value_LE_MaleAtBirth
##   <chr>     <chr>                                 <dbl>                <dbl>
## 1 E07000246 Somerset West and Taunton              19.5                 80.2
## 2 E07000246 Somerset West and Taunton              19.5                 80.2
## 3 E07000245 West Suffolk                           19.9                 81.1
## 4 E07000245 West Suffolk                           19.9                 81.1
## 5 E07000244 East Suffolk                           19.7                 80.4
## 6 E07000244 East Suffolk                           19.7                 80.4
```

Finally, select unique rows as we have generated duplicated entries in the previous step

```
lifeexpect = lifeexpect %>% distinct()
```

```
lifeexpect_time = updateCodes(lifeexpect_time)

combs_dup = lifeexpect_time %>%
  group_by(areaCode, areaName, year) %>% summarise(duplicat = n()>1)

print(
  lifeexpect_time[lifeexpect_time$areaCode %in%
               combs_dup$areaCode[combs_dup$duplicat==T],c(3,6,7)])
```

For lifeexpecttime

```
## # A tibble: 18 x 3
##    areaName                  Value_LE_MaleAt65 Value_LE_MaleAtBirth
##    <chr>                                 <dbl>                <dbl>
##  1 Somerset West and Taunton              18.9                 79.6
##  2 Somerset West and Taunton              18.9                 79.4
##  3 Somerset West and Taunton              18.9                 79.4
##  4 Somerset West and Taunton              19.4                 80.5
##  5 Somerset West and Taunton              20.0                 81.0
##  6 Somerset West and Taunton              20.0                 81.0
##  7 West Suffolk                             19.2                 80.5
##  8 West Suffolk                             19.5                 80.6
##  9 West Suffolk                             19.5                 80.6
## 10 West Suffolk                             20.1                 81.9
## 11 West Suffolk                             20.3                 81.6
## 12 West Suffolk                             20.3                 81.6
## 13 East Suffolk                             19.8                 81.4
## 14 East Suffolk                             20.2                 81.8
## 15 East Suffolk                             20.2                 81.8
## 16 East Suffolk                             19.1                 79.4
## 17 East Suffolk                             19.1                 79.0
## 18 East Suffolk                             19.1                 79.0
```

This function combines entries by taking the mean when they are duplicated

```
for (row in which(combs_dup$duplicat==T)){
  entry = combs_dup[row,]

  lifeexpect_time[(lifeexpect_time$areaCode == entry$areaCode &
                 lifeexpect_time$areaName == entry$areaName &
                 lifeexpect_time$year == entry$year),
              !colnames(lifeexpect_time) %in% c('areaCode','areaName', 'year')] =
    t(apply(lifeexpect_time[(lifeexpect_time$areaCode == entry$areaCode &
                           lifeexpect_time$areaName == entry$areaName &
                           lifeexpect_time$year == entry$year),
                        !colnames(lifeexpect_time) %in% c('areaCode','areaName', 'year')],2,mean))

}

print(
  lifeexpect_time[lifeexpect_time$areaCode %in%
               combs_dup$areaCode[combs_dup$duplicat==T],c(3,6,7)])
```

```
## # A tibble: 18 x 3
##    areaName                  Value_LE_MaleAt65 Value_LE_MaleAtBirth
##    <chr>                                 <dbl>                <dbl>
##  1 Somerset West and Taunton             19.2                 80.1
##  2 Somerset West and Taunton             19.5                 80.2
##  3 Somerset West and Taunton             19.5                 80.2
##  4 Somerset West and Taunton             19.2                 80.1
##  5 Somerset West and Taunton             19.5                 80.2
##  6 Somerset West and Taunton             19.5                 80.2
##  7 West Suffolk                          19.7                 81.2
##  8 West Suffolk                          19.9                 81.1
##  9 West Suffolk                          19.9                 81.1
## 10 West Suffolk                          19.7                 81.2
## 11 West Suffolk                          19.9                 81.1
## 12 West Suffolk                          19.9                 81.1
## 13 East Suffolk                          19.5                 80.4
## 14 East Suffolk                          19.7                 80.4
## 15 East Suffolk                          19.7                 80.4
## 16 East Suffolk                          19.5                 80.4
## 17 East Suffolk                          19.7                 80.4
## 18 East Suffolk                          19.7                 80.4
```

Finally, select unique rows as we have generated duplicated entries in the previous step

```
lifeexpect_time = lifeexpect_time %>% distinct()
```

Population table

Out if this dataset we will extract information regarding age statistics and population size

```
tail(population)
```

```
##           v4_0 calendar.years time admin.geography     geography
## 1897771 194462           2018 2018       E06000049 Cheshire East
## 1897772  56865           2018 2018       E07000112       Shepway
## 1897773  48444           2018 2018       E07000142  West Lindsey
## 1897774  42731           2018 2018       E07000236      Redditch
## 1897775  51607           2018 2018       E07000237     Worcester
## 1897776  48885           2018 2018       S12000030      Stirling
##         mid.year.pop.sex    sex mid.year.pop.age   age
## 1897771                2 Female            total Total
## 1897772                2 Female            total Total
## 1897773                2 Female            total Total
## 1897774                2 Female            total Total
## 1897775                2 Female            total Total
## 1897776                2 Female            total Total
```

We notice below that the column containing the population estimates contains no missing values which is great

```
sum(is.na(population$v4_0)) # no missing values, thank god, no need for imputation
```

```
## [1] 0
```

```
population = population %>%
  select(-c(calendar.years,
          mid.year.pop.sex,
          mid.year.pop.age))%>%
  rename(areaName = geography,
         areaCode = admin.geography,
         Value =  v4_0,
         year = time)
```

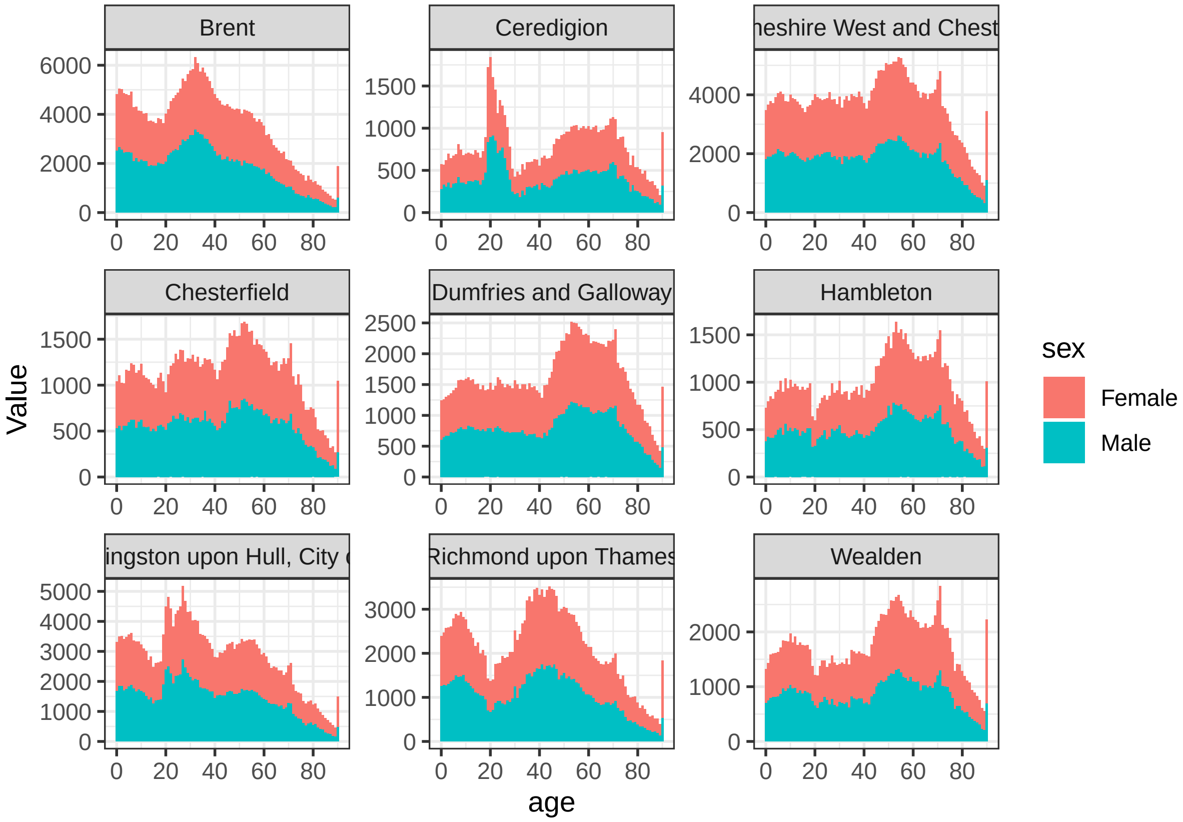
(un)select, filter, rename

Update old codes In this case we update the codes before tidying because we are going to perform aggregations later

```
population = updateCodes(population)
```

Visualise the age distribution Distribution of age coloured by gender for 9 random local authority districts

```
population %>%
  filter(year == 2018) %>%
  filter(sex != 'All',
         age != 'Total',
         areaName %in% sample(areaName, 9)) %>%
  mutate(age = as.numeric(ifelse(age == '90+', 90, age))) %>%
  ggplot(aes(x = age,
             y = Value,
             fill = sex))+
  geom_col()+
  facet_wrap(vars(areaName), scales = 'free')+
  scale_x_continuous(breaks = seq(0,100, by = 20))
```



Check if all ages are represented Check if there are any holes in the age records

```
unique_ages = data.matrix(unique(population %>%
                                  filter(age != 'Total') %>%
                                  mutate(age = as.numeric(ifelse(age == '90+', 90, age))) %>%
                                  select(age)))
all(unique_ages %in% c(0:90))
```

```
## [1] TRUE
```

For population I can extract: * age info: * mean * median * mode: when calculating the mode, sometimes for some miracle of the universe the frequency for a given age group and a given sex is the same (See Wrexham in population df as they have 1015 for 46 and 47 y/o males), in these cases I'm taking the mean for those cases (i.e. 46.5 in above example) * stdev * ~~range~~ * ~~max~~ * ~~min~~ in practice all LADs have at least one 0 year old and one 90+ year old * age bins, number of people in each age bin

- pop info:
  - total
  - total males
  - total females

All stratified by gender and not stratified

We define the age bins arbitrarily to represent: children and teens (0-16 y/o), young adults (17-29 y/o), adults (30-63 y/o), old (64-80) and very old (80+).

To calculate things like median and mode I actually ‘explode’ the age column with the `rep()` function getting the count for each age from the `Value` column and then calculate the statistics in question

```
age_time = population %>%
  filter(age != 'Total' & sex != 'All') %>%
  mutate(age = as.numeric(ifelse(age == '90+', 90, age))) %>%
  group_by(year,
           areaCode,
           areaName,
           sex) %>%
  summarise(TotalPeople = sum(Value),
            MeanAge = sum(age*Value)/TotalPeople,
            MedianAge = median(rep(age,Value)),
            ModeAge = mean(age[which(Value == max(Value))]),
            stdevAge = sd(rep(age,Value)),
            NumberOfTeens = sum(Value[which(age<17)]),
            NumberOfYoungAdults = sum(Value[which(age>=17 & age<30)]),
            NumberOfAdults = sum(Value[which(age>=30 & age<64)]),
            NumberOfOld = sum(Value[which(age>=64 & age<81)]),
            NumberOfVeryOld = sum(Value[which(age>80)]))

age = age_time %>% filter(year == 2018)

age_time = age_time %>%
  pivot_wider(names_from = sex,
              values_from = c(TotalPeople,
                              MeanAge,
                              MedianAge,
                              ModeAge,
                              stdevAge,
                              NumberOfTeens,
                              NumberOfYoungAdults,
                              NumberOfAdults,
                              NumberOfOld,
                              NumberOfVeryOld)
  )


age  = age %>%
  pivot_wider(names_from = sex,
              values_from = c(TotalPeople,
                              MeanAge,
                              MedianAge,
                              ModeAge,
                              stdevAge,
                              NumberOfTeens,
                              NumberOfYoungAdults,
                              NumberOfAdults,
                              NumberOfOld,
                              NumberOfVeryOld)
  )
```

```
agetotal_time = population %>%
  filter(age != 'Total' & sex != 'All') %>%
  mutate(age = as.numeric(ifelse(age == '90+', 90, age))) %>%
  group_by(year,
           areaCode,
           areaName) %>% # note we are not aggregating by sex
  summarise(TotalPeople = sum(Value),
            MeanAge = sum(age*Value)/TotalPeople,
            MedianAge = median(rep(age,Value)),
            ModeAge = mean(age[which(Value == max(Value))]),
            stdevAge = sd(rep(age,Value)),
            NumberOfTeens = sum(Value[which(age<17)]),
            NumberOfYoungAdults = sum(Value[which(age>=17 & age<30)]),
            NumberOfAdults = sum(Value[which(age>=30 & age<64)]),
            NumberOfOld = sum(Value[which(age>=64 & age<81)]),
            NumberOfVeryOld = sum(Value[which(age>80)]))

agetotal = agetotal_time %>% filter(year == 2018)

colnames(agetotal_time)[4:ncol(agetotal_time)] = paste0(
  colnames(agetotal_time)[4:ncol(agetotal_time)], '_All')

colnames(agetotal)[4:ncol(agetotal)] = paste0(
  colnames(agetotal)[4:ncol(agetotal)], '_All')
```

Total(Female+Male) population statistics

```
popclean_time = merge(age_time, agetotal_time, by = c('year','areaCode','areaName'))

popclean = merge(age, agetotal, by = c('year','areaCode','areaName'))
```

Merge into a single table

Update old area codes Pass through custom function to update old codes, then combine the entries with matching area name and area code

```
popclean = updateCodes(popclean)

combs_dup = popclean %>%
  group_by(areaCode, areaName,year) %>% summarise(duplicat = n()>1)

# In this case luckily we had no bad entries
print(
  popclean[popclean$areaCode %in%
             combs_dup$areaCode[combs_dup$duplicat==T],c(3,6,7)])
```

For population table

```
## [1] areaName       MeanAge_Female MeanAge_Male
## <0 rows> (or 0-length row.names)
```

```
popclean_time = updateCodes(popclean_time)

combs_dup = popclean_time %>%
  group_by(areaCode, areaName,year) %>% summarise(duplicat = n()>1)

print(
  popclean_time[popclean_time$areaCode %in%
               combs_dup$areaCode[combs_dup$duplicat==T],c(3,6,7)])
```

For popclean_time table

```
## [1] areaName       MeanAge_Female MeanAge_Male
## <0 rows> (or 0-length row.names)
```

Suicides table

```
str(suicides)
```

```
## 'data.frame':    6392 obs. of  5 variables:
##  $ V4_0           : int  25 20 15 6 13 10 11 50 358 21 ...
##  $ calendar.years : int  2015 2013 2005 2016 2010 2004 2012 2011 2006 2008 ...
##  $ time           : int  2015 2013 2005 2016 2010 2004 2012 2011 2006 2008 ...
##  $ admin.geography: chr  "E08000022" "E06000009" "E08000021" "E07000037" ...
##  $ geography      : chr  "North Tyneside" "Blackpool" "Newcastle upon Tyne" "High Peak" ...
```

```
suicides = suicides %>%
  select(-calendar.years) %>%
  rename(Value = V4_0,
         year = time,
         areaCode = admin.geography,
         areaName = geography)
```

Select, rename

```
sum(is.na(suicides$Value)) # no NAs good
```

Again no NAs

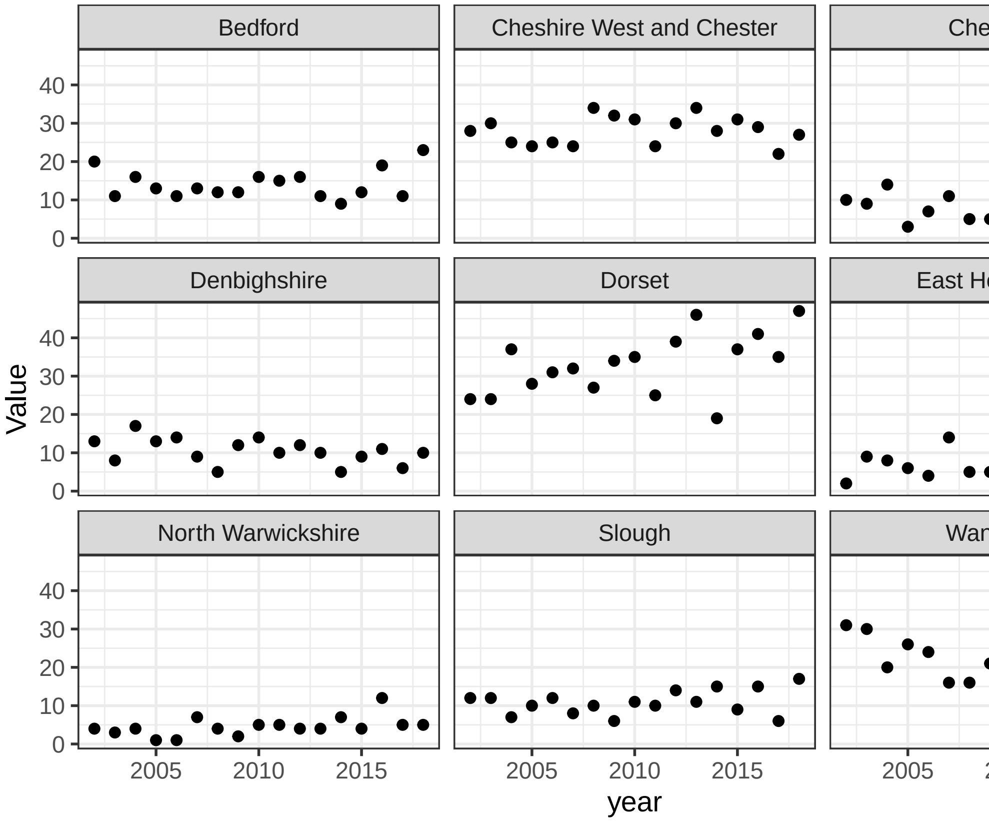
```
## [1] 0
```

According the command below all locations have data available for all years

```
sum(is.na(suicides %>%
            pivot_wider(names_from = year, values_from = Value)))
```

```
## [1] 0
```

```
suicides %>%
  filter(areaName %in% sample(areaName, 9)) %>%
  ggplot(aes(x = year, y = Value))+
  geom_point()+
  facet_wrap(vars(areaName))
```



Time-series plot of a few LADs

```
suicides = updateCodes(suicides)

combs_dup = suicides %>%
  group_by(areaCode, areaName, year) %>% summarise(duplicat = n()>1)

print(
  suicides[suicides$areaCode %in%
             combs_dup$areaCode[combs_dup$duplicat==T],c(1)])
```

Update codes

```
## integer(0)
```

No duplicates again!

```
suicides_time = suicides  %>% rename(NumberOfSuicides = Value)
suicides = suicides_time %>% filter(year == 2018)
```

Store time-series and most up-to-date one

Wellbeing table

```r
tail(wellbeing)
```

Evaluate content of table

```
##       V4_3 Data.Marking Lower.limit Upper.limit yyyy.yy    time admin.geography
## 69115 2.99                     2.73        3.25 2013-14 2013-14       S12000019
## 69116 3.02                     2.79        3.26 2017-18 2017-18       S12000021
## 69117 2.69                     2.44        2.94 2016-17 2016-17       S12000024
## 69118 2.33                     2.11        2.55 2016-17 2016-17       S12000026
## 69119 3.12                     2.88        3.35 2012-13 2012-13       S12000039
## 69120 2.07                     1.61        2.53 2015-16 2015-16       N09000001
##                     geography wellbeing.measureofwellbeing
## 69115              Midlothian                      anxiety
## 69116          North Ayrshire                      anxiety
## 69117       Perth and Kinross                      anxiety
## 69118        Scottish Borders                      anxiety
## 69119     West Dunbartonshire                      anxiety
## 69120 Antrim and Newtownabbey                      anxiety
##       allmeasuresofwellbeing wellbeing.estimate       estimate
## 69115                Anxiety       average-mean Average (mean)
## 69116                Anxiety       average-mean Average (mean)
## 69117                Anxiety       average-mean Average (mean)
## 69118                Anxiety       average-mean Average (mean)
## 69119                Anxiety       average-mean Average (mean)
## 69120                Anxiety       average-mean Average (mean)
```

Missing data We observe quite a lot of data missingness here

```r
sum(is.na(wellbeing$V4_3))
```

```
## [1] 15914
```

```r
mean(is.na(wellbeing$V4_3))
```

```
## [1] 0.2302373
```

```r
wellbeing = wellbeing %>%
  select(-c(Data.Marking,
            yyyy.yy,
            wellbeing.measureofwellbeing,
            wellbeing.estimate)) %>%
  rename(Value = V4_3,
         lowerCI = Lower.limit,
         upperCI = Upper.limit,
         year = time,
         areaCode = admin.geography,
         areaName = geography,
         WellbeingMetric = allmeasuresofwellbeing,
         EstimateBin = estimate) %>%
  mutate(lowerCI = as.numeric(lowerCI),
         upperCI = as.numeric(upperCI))
```

Select, rename, mutate

Turn year interval to year Same operation that we did for the `lifeexpect` table, here the intervals are only 2 years long though. (e.g. 2014-15 –> 2015)

```
wellbeing$year = as.numeric(lapply(strsplit(wellbeing$year, '-'), '[[',1))+1
```

Variable selection The wellbeing data is very rich, this data was gathered by running a survey with 4 self-assesed questions. For each question the user can score a value between 0 and 10. The `wellbeing` table contains the average score for each category (i.e. `r{paste(unique(wellbeing$wellbeing.measureofwellbeing), collapse= ', ')}`) as well as the proportion of people in different bins (i.e. `r{paste(unique(wellbeing$estimate), collapse= ', ')}`). I believe the average score is more interpretable than the bins and more concise, though there are some caveats to this. Read more about the methodology behing these metrics in the ONS site

```
wellbeing = wellbeing %>% filter(EstimateBin == 'Average (mean)')
```

```
sum(is.na(wellbeing$Value)) # 117 missing values
```

Evaluate missingness after filtering

```
## [1] 117
```

Missingness by area and by wellbeing metric In this case if the output is 8, all entries are missing for all 8 years the variable was measured

```
wellbeing %>%
  group_by(areaCode,
           WellbeingMetric) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value)),
            ProportionOfMissingEntries = mean(is.na(Value))) %>%
  group_by(NumberOfMissingEntries,
           ProportionOfMissingEntries) %>%
  summarise(Count = n()) %>%
  kable()
```

NumberOfMissingEntries	ProportionOfMissingEntries	Count
0	0.000	1668
1	0.125	51
2	0.250	1
8	1.000	8

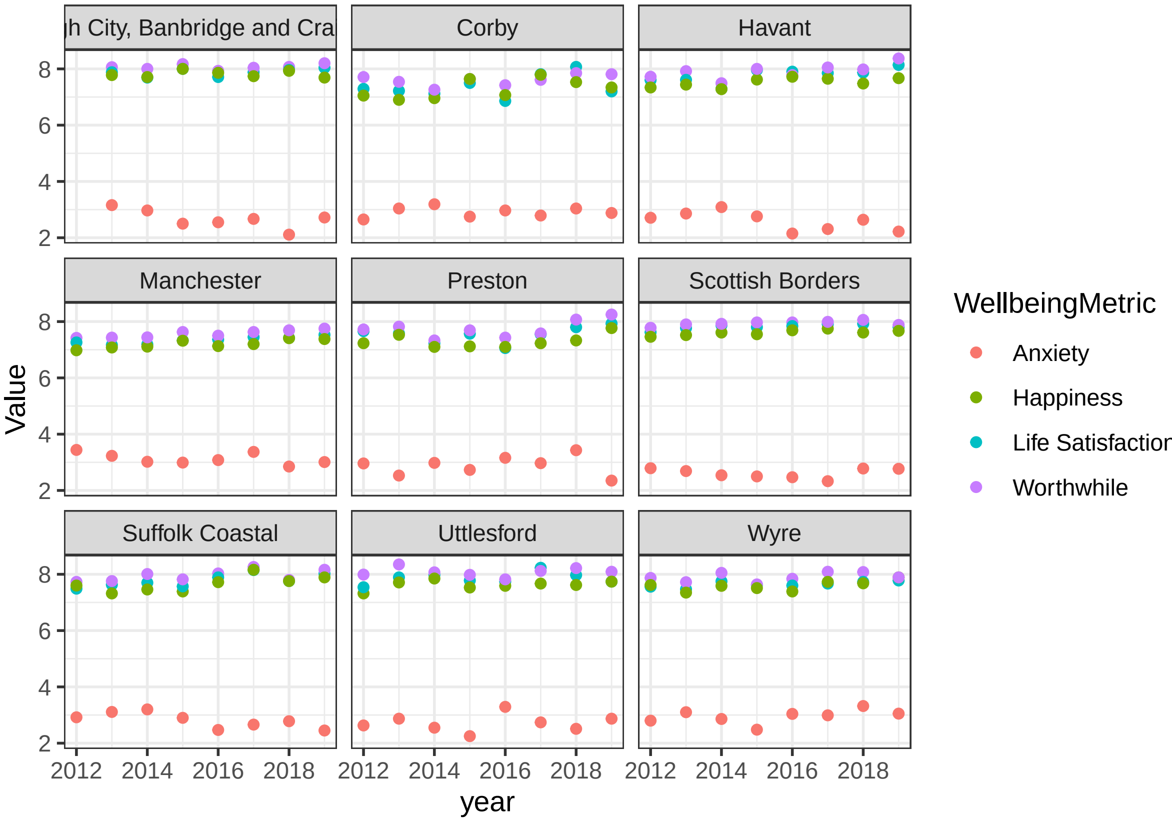
```
wellbeing %>%
  group_by(year) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value)),
            ProportionOfMissingEntries = mean(is.na(Value))) %>%
  kable()
```

Missingness by year

year	NumberOfMissingEntries	ProportionOfMissingEntries
2012	52	0.0300926
2013	8	0.0046296
2014	8	0.0046296
2015	9	0.0052083

year	NumberOfMissingEntries	ProportionOfMissingEntries
2016	8	0.0046296
2017	8	0.0046296
2018	12	0.0069444
2019	12	0.0069444

```
wellbeing %>%
  filter(areaName %in% sample(areaName, 9)) %>%
  ggplot(aes(x = year,
             y = Value,
             color = WellbeingMetric)) +
  geom_point() +
  facet_wrap(vars(areaName))
```



Time-series plot

Edge-cases The LADs with all missingness are the City of london and the Isles of Scilly. These two often have no available data from the ONS #### Time-series imputation Order the data before imputation (just in case)

```
wellbeing = wellbeing %>% arrange(areaCode, WellbeingMetric, year)

wellbeing = time_impute(wellbeing, areaCode, WellbeingMetric)
```

```
## [1] "Missingness before imputation (%):"
##       Value     lowerCI     upperCI
## 0.008463542 0.008463542 0.008463542
## Time difference of 18.52285 secs
## [1] "Missingness after imputation (%):"
##       Value     lowerCI     upperCI
## 0.004629630 0.008463542 0.008463542
```

```
wellbeing %>%
  group_by(areaCode,
           WellbeingMetric) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value)),
            ProportionOfMissingEntries = mean(is.na(Value))) %>%
  group_by(NumberOfMissingEntries,
           ProportionOfMissingEntries) %>%
  summarise(Count = n()) %>%
  kable()
```

Results (as expected)

NumberOfMissingEntries	ProportionOfMissingEntries	Count
0	0	1720
8	1	8

(Updated) missingness by year

```
wellbeing %>%
  group_by(year) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value))) %>%
  kable()
```

year	NumberOfMissingEntries
2012	8
2013	8
2014	8
2015	8
2016	8
2017	8
2018	8
2019	8

```
wellbeing_time = wellbeing %>%
  mutate(Range = upperCI - lowerCI) %>%
  select(Value,
         Range,
         year,
         areaCode,
         areaName,
         WellbeingMetric) %>%
  pivot_wider(names_from = WellbeingMetric,
              values_from = c(Value, Range))
```

Gather data

```
wellbeing_time = updateCodes(wellbeing_time)

combs_dup = wellbeing_time %>%
  group_by(areaCode, areaName, year) %>% summarise(duplicat = n()>1)

print(
  wellbeing_time[wellbeing_time$areaCode %in%
              combs_dup$areaCode[combs_dup$duplicat==T],c(1,2,3,4)] %>%
    arrange(areaCode, areaName,year), n = 100)
```

Update codes

```
## # A tibble: 48 x 4
##     year areaCode  areaName                  Value_Anxiety
##    <dbl> <chr>     <chr>                             <dbl>
##  1  2012 E07000244 East Suffolk                       2.92
##  2  2012 E07000244 East Suffolk                       3.27
##  3  2013 E07000244 East Suffolk                       3.11
##  4  2013 E07000244 East Suffolk                       2.51
##  5  2014 E07000244 East Suffolk                       3.2
##  6  2014 E07000244 East Suffolk                       2.96
##  7  2015 E07000244 East Suffolk                       2.9
##  8  2015 E07000244 East Suffolk                       3.13
##  9  2016 E07000244 East Suffolk                       2.47
## 10  2016 E07000244 East Suffolk                       2.81
## 11  2017 E07000244 East Suffolk                       2.66
## 12  2017 E07000244 East Suffolk                       2.78
## 13  2018 E07000244 East Suffolk                       2.78
## 14  2018 E07000244 East Suffolk                       2.69
## 15  2019 E07000244 East Suffolk                       2.45
## 16  2019 E07000244 East Suffolk                       3.1
## 17  2012 E07000245 West Suffolk                       2.74
## 18  2012 E07000245 West Suffolk                       3.18
## 19  2013 E07000245 West Suffolk                       2.72
## 20  2013 E07000245 West Suffolk                       3.1
## 21  2014 E07000245 West Suffolk                       3.14
## 22  2014 E07000245 West Suffolk                       2.58
## 23  2015 E07000245 West Suffolk                       3.51
## 24  2015 E07000245 West Suffolk                       2.47
## 25  2016 E07000245 West Suffolk                       2.81
## 26  2016 E07000245 West Suffolk                       3.12
## 27  2017 E07000245 West Suffolk                       3.32
## 28  2017 E07000245 West Suffolk                       2.29
## 29  2018 E07000245 West Suffolk                       2.77
## 30  2018 E07000245 West Suffolk                       2.63
## 31  2019 E07000245 West Suffolk                       2.46
## 32  2019 E07000245 West Suffolk                       2.23
## 33  2012 E07000246 Somerset West and Taunton          2.79
## 34  2012 E07000246 Somerset West and Taunton          3.31
## 35  2013 E07000246 Somerset West and Taunton          3.24
## 36  2013 E07000246 Somerset West and Taunton          3.08
## 37  2014 E07000246 Somerset West and Taunton          3.1
## 38  2014 E07000246 Somerset West and Taunton          2.19
## 39  2015 E07000246 Somerset West and Taunton          2.97
## 40  2015 E07000246 Somerset West and Taunton          2.55
## 41  2016 E07000246 Somerset West and Taunton          2.56
## 42  2016 E07000246 Somerset West and Taunton          2.92
## 43  2017 E07000246 Somerset West and Taunton          3.1
## 44  2017 E07000246 Somerset West and Taunton          2.3
## 45  2018 E07000246 Somerset West and Taunton          2.46
## 46  2018 E07000246 Somerset West and Taunton          2.67
## 47  2019 E07000246 Somerset West and Taunton          2.65
## 48  2019 E07000246 Somerset West and Taunton          2.67
```

Many duplicates now

#' This function combines entries by taking the **mean** when they are duplicated

```
for (row in which(combs_dup$duplicat==T)){
  entry = combs_dup[row,]

  wellbeing_time[(wellbeing_time$areaCode == entry$areaCode &
                 wellbeing_time$areaName == entry$areaName &
                 wellbeing_time$year == entry$year),
              !colnames(wellbeing_time) %in% c('areaCode','areaName', 'year')] =
    t(apply(wellbeing_time[(wellbeing_time$areaCode == entry$areaCode &
                           wellbeing_time$areaName == entry$areaName &
                           wellbeing_time$year == entry$year),
                        !colnames(wellbeing_time) %in% c('areaCode','areaName', 'year')],2,mean))

}

print(
  wellbeing_time[wellbeing_time$areaCode %in%
                combs_dup$areaCode[combs_dup$duplicat==T],c(1,2,3,4)])
```

```
## # A tibble: 48 x 4
##      year areaCode  areaName                  Value_Anxiety
##     <dbl> <chr>     <chr>                             <dbl>
##   1  2012 E07000246 Somerset West and Taunton          3.05
##   2  2013 E07000246 Somerset West and Taunton          3.16
##   3  2014 E07000246 Somerset West and Taunton          2.64
##   4  2015 E07000246 Somerset West and Taunton          2.76
##   5  2016 E07000246 Somerset West and Taunton          2.74
##   6  2017 E07000246 Somerset West and Taunton          2.7
##   7  2018 E07000246 Somerset West and Taunton          2.56
##   8  2019 E07000246 Somerset West and Taunton          2.66
##   9  2012 E07000246 Somerset West and Taunton          3.05
##  10  2013 E07000246 Somerset West and Taunton          3.16
## # ... with 38 more rows
```

Finally, select unique rows as we have generated duplicated entries in the previous step

```
wellbeing_time = wellbeing_time %>% distinct()

wellbeing = wellbeing_time %>%
  filter(year == 2019)
```

Work table

The ASHE work table contains a column called coefficient of variation which is an estimate of the variation within the sample and it helps evaluate whether the sample can be considered a good representation of the population. Find explanation of the Coefficient of variation (CV) column here Moreover, in this section we have two tables, `work` and `work_ts`, the 1st one contains info for 2019 and the second one contains data from earlier years to enable us to impute the 2019 data.

```
str(work)
```

```
## 'data.frame':    1002672 obs. of  17 variables:
##  $ V4_2                       : num  27.5 38.4 37.5 37.5 39.9 40.1 36 35 NA 20.3 ...
##  $ Data.Marking               : chr  "" "" "" "" ...
##  $ CV                         : chr  "0.1" "1.7" "0.0" "0.1" ...
##  $ calendar.years             : int  2019 2019 2019 2019 2019 2019 2019 2019 2019 2019 ...
##  $ time                       : int  2019 2019 2019 2019 2019 2019 2019 2019 2019 2019 ...
##  $ admin.geography            : chr  "K02000001" "E06000002" "K02000001" "E92000001" ...
##  $ geography                  : chr  "United Kingdom" "Middlesbrough" "United Kingdom" "England" ...
##  $ ashe.statistics            : chr  "25" "80" "70" "70" ...
##  $ statistics                 : chr  "25th percentile" "80th percentile" "70th percentile" "70th per
##  $ ashe.sex                   : chr  "all" "all" "all" "all" ...
##  $ sex                        : chr  "All" "All" "All" "All" ...
##  $ ashe.working.pattern       : chr  "all" "all" "all" "all" ...
##  $ workingpattern             : chr  "All" "All" "All" "All" ...
##  $ ashe.hours.and.earnings    : chr  "paid-hours-worked-basic" "paid-hours-worked-basic" "paid-hours-
##  $ hoursandearnings           : chr  "Paid hours worked - Basic" "Paid hours worked - Basic" "Paid h
##  $ ashe.workplace.or.residence: chr  "workplace" "workplace" "workplace" "workplace" ...
##  $ workplaceorresidence       : chr  "Workplace" "Workplace" "Workplace" "Workplace" ...
```

```
work = work %>%
  select(-c(Data.Marking,
            calendar.years,
            ashe.working.pattern,
            ashe.sex,
            ashe.statistics,
            ashe.workplace.or.residence,
            ashe.hours.and.earnings,
            ashe.working.pattern)) %>%
  rename(Value = V4_2,
         year = time,
         areaCode = admin.geography,
         areaName = geography,
         CoefficientOfVariation = CV) %>%
  mutate(CoefficientOfVariation = as.numeric(CoefficientOfVariation))
```

Select, rename, mutate

```
unique(work$hoursandearnings)
```

Explore a few columns

```
## [1] "Paid hours worked - Basic"       "Paid hours worked - Overtime"
## [3] "Weekly pay - Excluding overtime" "Basic pay - Including other pay"
## [5] "Weekly pay - Gross"              "Overtime pay"
##  [7] "Hourly pay - Gross"              "Annual pay - Gross"
##  [9] "Hourly pay - Excluding overtime" "Annual pay - Incentive"
## [11] "Paid hours worked - Total"
```

```
unique(work$workplaceorresidence)
```

```
## [1] "Workplace" "Residence"
```

```
unique(work$workingpattern)
```

```
## [1] "All"       "Full-Time" "Part-Time"
```

```
unique(work$statistics)
```

```
##  [1] "25th percentile" "80th percentile" "70th percentile" "90th percentile"
##  [5] "30th percentile" "40th percentile" "60th percentile" "20th percentile"
##  [9] "Median"          "75th percentile" "10th percentile" "Mean"
```

Many, many, many statistics available Too many stats are available as seen above, we are only interested in a few. We select mean, median, 10th and 90th percentile

```
work = work %>%
  filter(statistics %in% c('Mean', 'Median', '10th percentile', '90th percentile'),
         workplaceorresidence == 'Workplace')
```

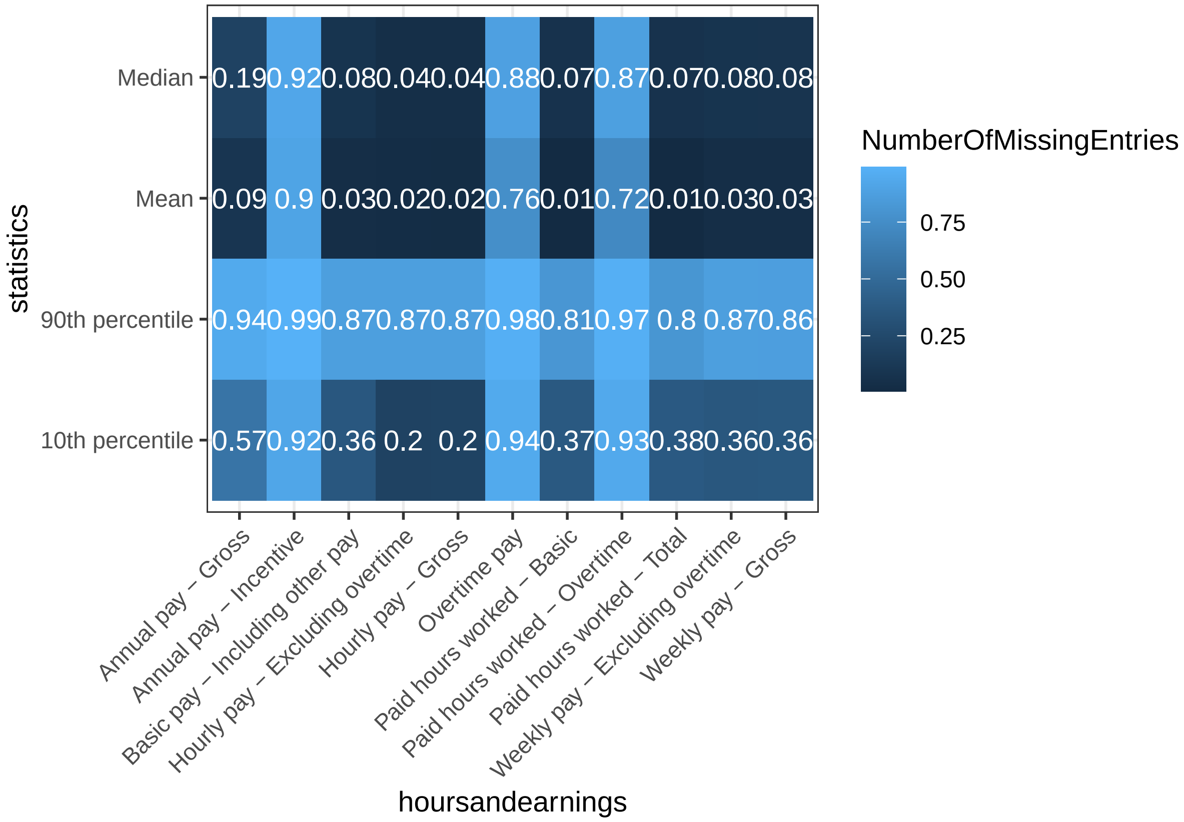
Select earnings metric From the above command we see that hourly pay has the least missingness so we'll go with that one

```
work %>%
  group_by(hoursandearnings) %>%
  summarise(NumberOfMissingEntries = mean(is.na(Value))) %>%
  kable()
```

hoursandearnings	NumberOfMissingEntries
Annual pay - Gross	0.4504344
Annual pay - Incentive	0.9311480
Basic pay - Including other pay	0.3344523
Hourly pay - Excluding overtime	0.2819905
Hourly pay - Gross	0.2815298
Overtime pay	0.8890864
Paid hours worked - Basic	0.3138494
Paid hours worked - Overtime	0.8761190
Paid hours worked - Total	0.3143102
Weekly pay - Excluding overtime	0.3342549
Weekly pay - Gross	0.3350448

Now we evaulate the missingess of each earning metric per statistic to see if some combination of these is missing at an unacceptable rate

```
work %>%
  group_by(hoursandearnings, statistics) %>%
  summarise(NumberOfMissingEntries = mean(is.na(Value))) %>%
  ggplot(aes(x = hoursandearnings,
             y = statistics,
             fill = NumberOfMissingEntries,
             label = round(NumberOfMissingEntries,2)))+
  geom_tile()+
  geom_text(color = 'white')+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



this last graph reveals that the 90th and the 10th percentile stats have too much missingess hence I am dropping them, it also show how Hourly pay is the most covered stat

Based on previous graph I decided to choose hourly pay and mean and median only

```
work = work %>%
  filter(hoursandearnings == 'Hourly pay - Gross',
         statistics %in% c('Median', 'Mean'))
```

Given that now the `hoursandearnings` and the `worplaceorresidence` columns contain unique values only I drop them

```
work = work %>% select(-c(hoursandearnings, workplaceorresidence))
```

Work time-series table

```
work_ts = work_ts %>%
  select(-c(Data_Marking,
            calendar.years,
            ashe.working.pattern,
            ashe.sex,
            ashe.statistics,
            ashe.workplace.or.residence,
            ashe.hours.and.earnings,
            ashe.working.pattern)) %>%
  rename(Value = V4_2,
         year = time,
         areaCode = admin.geography,
         areaName = geography,
         CoefficientOfVariation = CV) %>%
  mutate(CoefficientOfVariation = as.numeric(CoefficientOfVariation))
```

Select, rename, mutate Filter the same way as for the `work` dataframe

```
work_ts = work_ts %>%
  filter(statistics %in% c('Mean', 'Median', '10th percentile', '90th percentile'),
         workplaceorresidence == 'Workplace')

work_ts = work_ts %>%
  filter(hoursandearnings == 'Hourly pay - Gross',
         statistics %in% c('Median', 'Mean'))

work_ts = work_ts %>% select(-c(hoursandearnings, workplaceorresidence))
```

```

unique(work$areaName[!(work$areaName %in% work_ts$areaName)])
```

Check for any disparity between the areas covered by both of these dataframes

```
## [1] "East Suffolk"                        "West Suffolk"
## [3] "Bournemouth, Christchurch and Poole" "Somerset West and Taunton"
```

Now merge work (data from 2019 only) and work_ts (2016 through 2018)

```
work = rbind(work, work_ts)
```

Start exploring dataset I believe not all towns have entries for all years in theory there should be 4 entries (2016-19) per combination of areacode, stat, sex and working pattern

```
work %>%
  group_by(areaCode,
           statistics,
           sex,
           workingpattern) %>%
  summarise(NumberOfYearsWithAvailableData = n()) %>% ## this takes the numbers
  # of entries per unique combination of the variables in the group_by
  group_by(NumberOfYearsWithAvailableData) %>%
  summarise(Count = n()) %>%
  kable()
```

NumberOfYearsWithAvailableData	Count
1	126
3	306
4	7470

As you can see in the above table for some combinations there are only 3 available variables hence 3 missing variables represent 100% where as in other cases 4 missing varialbes represent 100% of variables

```
work %>%
  group_by(areaCode,
           statistics,
           sex,
           workingpattern) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value)),
            ProportionOfMissingEntries = mean(is.na(Value))) %>%
  group_by(NumberOfMissingEntries,
           ProportionOfMissingEntries) %>%
  summarise(Count = n()) %>%
  kable()
```

NumberOfMissingEntries	ProportionOfMissingEntries	Count
0	0.0000000	7384
1	0.2500000	213
1	0.3333333	7
1	1.0000000	1
2	0.5000000	138
2	0.6666667	13
3	0.7500000	81
3	1.0000000	6
4	1.0000000	59

Missingness by year

```
work %>%
  group_by(year) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value)),
            ProportionOfMissingEntries = mean(is.na(Value))) %>%
  kable()
```

year	NumberOfMissingEntries	ProportionOfMissingEntries
2016	266	0.0342078
2017	258	0.0331790
2018	265	0.0340792
2019	231	0.0304107

Time-series imputation Make sure data is ordered by year

```
work = work %>% arrange(areaCode, statistics, sex, workingpattern, year)
work = time_impute(work, cols_to_impute = c('Value', 'CoefficientOfVariation'),
                   areaCode, statistics, sex, workingpattern)
```

```
## [1] "Missingness before imputation (%):"
##                  Value CoefficientOfVariation
##             0.03298409             0.01442246
## Time difference of 1.859533 mins
## [1] "Missingness after imputation (%):"
##                  Value CoefficientOfVariation
##            0.008246023            0.014422455
```

```
work %>%
  group_by(areaCode,
           statistics,
           sex,
           workingpattern) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value)),
            ProportionOfMissingEntries = mean(is.na(Value))) %>%
  group_by(NumberOfMissingEntries,
           ProportionOfMissingEntries) %>%
  summarise(Count = n())
```

Results: (updated) missingness

```
## # A tibble: 4 x 3
## # Groups:   NumberOfMissingEntries [4]
##   NumberOfMissingEntries ProportionOfMissingEntries Count
##                    <int>                      <dbl> <int>
## 1                      0                          0  7836
## 2                      1                          1     1
## 3                      3                          1     6
## 4                      4                          1    59
```

```
work %>%
  group_by(year) %>%
  summarise(NumberOfMissingEntries = sum(is.na(Value)),
            ProportionOfMissingEntries = mean(is.na(Value))) %>%
  kable()
```

year	NumberOfMissingEntries	ProportionOfMissingEntries
2016	65	0.0083591
2017	65	0.0083591
2018	65	0.0083591
2019	60	0.0078989

Edge cases Finally we gotta be careful with those entries where records ends in 2018, we have to characterise them

```
work %>%
  group_by(areaName) %>%
  summarise(latestAvailableYear = max(year)) %>%
  group_by(latestAvailableYear) %>%
  summarise(Count = n()) %>%
  kable()
```

latestAvailableYear	Count
2018	14
2019	421

Note: interestingly, by area code you get slightly different numbers than grouping by areaName

```
work %>%
  group_by(areaCode) %>%
  summarise(latestAvailableYear = max(year)) %>%
  group_by(latestAvailableYear) %>%
  summarise(Count = n()) %>%
  kable()
```

latestAvailableYear	Count
2018	17
2019	422

Check count per number of available values

```
work %>%
  group_by(areaCode,
           areaName,
           statistics,
           sex,
           workingpattern) %>%
  summarise(NumberOfValues = n()) %>%
  group_by(NumberOfValues) %>%
  summarise(Count = n()) %>%
  kable()
```

NumberOfValues	Count
1	126
3	306
4	7470

Check also which area names have several latest years available

```
tst = work %>% group_by(areaCode, areaName) %>% summarise(latest = max(year))
tab = table(tst$areaName)
tab[tab>1]
```

```
##
##            Dorset      Glasgow City North Lanarkshire     West Midlands
##                 2                 2                 2                 2
```

Ok, so some codes have been updated which is fine, for example see Dorset code from 2018 corresponds to one that has been deprecated, what we'll do now is for each area name choose a single area code (the area code corresponding to the latest year)

```

work %>% group_by(areaCode, areaName) %>% summarise(year = max(year)) %>% filter(year == 2018)
```

Get entries for which latest available data is from 2019

```
## # A tibble: 17 x 3
## # Groups:   areaCode [17]
##    areaCode  areaName                year
##    <chr>     <chr>                  <int>
##  1 E06000028 Bournemouth             2018
##  2 E06000029 Poole                   2018
##  3 E07000048 Christchurch            2018
##  4 E07000049 East Dorset            2018
##  5 E07000050 North Dorset           2018
##  6 E07000051 Purbeck                2018
##  7 E07000052 West Dorset            2018
##  8 E07000053 Weymouth and Portland  2018
##  9 E07000190 Taunton Deane          2018
## 10 E07000191 West Somerset          2018
## 11 E07000201 Forest Heath           2018
## 12 E07000204 St Edmundsbury         2018
## 13 E07000205 Suffolk Coastal        2018
## 14 E07000206 Waveney                2018
## 15 E10000009 Dorset                 2018
## 16 S12000044 North Lanarkshire      2018
## 17 S12000046 Glasgow City           2018
```

it further appears that many areaName for which latest figures come from 2018 are so called 'abolished' now (i.e. inactive), see Suffolk Coastal or East Dorset So I have not check all the 14 entries which have missing data, the robust way to do this would be filtering with nspl (unclear what I meant in this comment)

```
length(unique(work$areaName))
```

```
## [1] 435
```

```
length(unique(work$areaCode))
```

```
## [1] 439
```

A final problem which is not really a problem is the fact that because the work table has data on regions(also UK-wide, England-wide…) as well as local authorities, If a region happens to be called the same than a LA then it'll appear twice (e.g. West Midlands E1200005 - Region code, E1100005 - LA code) #### Gather final data

```
work_time = work %>%
  rename(Pay = Value,
         CoV = CoefficientOfVariation) %>%
  pivot_wider(names_from = c(statistics,
                             workingpattern,
                             sex),
              values_from = c(Pay,
                              CoV))
```

```

work_time = updateCodes(work_time)

combs_dup = work_time %>%
  group_by(areaCode, areaName, year) %>% summarise(duplicat = n()>1)

print(
  work_time[work_time$areaCode %in%
                   combs_dup$areaCode[combs_dup$duplicat==T],c(1,2,3,4)] %>%
    arrange(areaCode, areaName,year), n = 100)
```

Update codes

```
## # A tibble: 21 x 4
##     year areaCode  areaName                   Pay_Mean_All_All
##    <int> <chr>     <chr>                                 <dbl>
##   1  2016 E07000244 East Suffolk                          15.1
##   2  2016 E07000244 East Suffolk                          12.1
##   3  2017 E07000244 East Suffolk                          15.1
##   4  2017 E07000244 East Suffolk                          12.3
##   5  2018 E07000244 East Suffolk                          14.1
##   6  2018 E07000244 East Suffolk                          13.5
##   7  2019 E07000244 East Suffolk                          15.0
##   8  2016 E07000245 West Suffolk                          13.0
##   9  2016 E07000245 West Suffolk                          13.8
##  10  2017 E07000245 West Suffolk                          13.9
##  11  2017 E07000245 West Suffolk                          13.7
##  12  2018 E07000245 West Suffolk                          13.6
##  13  2018 E07000245 West Suffolk                          14.4
##  14  2019 E07000245 West Suffolk                          14.9
##  15  2016 E07000246 Somerset West and Taunton             14.0
##  16  2016 E07000246 Somerset West and Taunton             16.0
##  17  2017 E07000246 Somerset West and Taunton             13.7
##  18  2017 E07000246 Somerset West and Taunton             16.3
##  19  2018 E07000246 Somerset West and Taunton             13.9
##  20  2018 E07000246 Somerset West and Taunton             17.3
##  21  2019 E07000246 Somerset West and Taunton             15.9
```

Many duplicates now

This function combines entries by taking the **mean** when they are duplicated

```
for (row in which(combs_dup$duplicat==T)){
  entry = combs_dup[row,]

  work_time[(work_time$areaCode == entry$areaCode &
                  work_time$areaName == entry$areaName &
                  work_time$year == entry$year),
               !colnames(work_time) %in% c('areaCode','areaName', 'year')] =
    t(apply(work_time[(work_time$areaCode == entry$areaCode &
                            work_time$areaName == entry$areaName &
                            work_time$year == entry$year),
                         !colnames(work_time) %in% c('areaCode','areaName', 'year')],2,mean))

}

print(
  work_time[work_time$areaCode %in%
                   combs_dup$areaCode[combs_dup$duplicat==T],c(1,2,3,4)])
```

```
## # A tibble: 21 x 4
##     year areaCode  areaName                  Pay_Mean_All_All
##    <int> <chr>     <chr>                                <dbl>
##   1  2016 E07000246 Somerset West and Taunton             15.0
##   2  2017 E07000246 Somerset West and Taunton             15.0
##   3  2018 E07000246 Somerset West and Taunton             15.6
##   4  2016 E07000246 Somerset West and Taunton             15.0
##   5  2017 E07000246 Somerset West and Taunton             15.0
##   6  2018 E07000246 Somerset West and Taunton             15.6
##   7  2016 E07000245 West Suffolk                          13.4
##   8  2017 E07000245 West Suffolk                          13.8
##   9  2018 E07000245 West Suffolk                          14.0
## 10  2016 E07000245 West Suffolk                             13.4
## # ... with 11 more rows
```

Finally, select unique rows as we have generated duplicated entries in the previous step

```
work_time = work_time %>% distinct()


work = work_time %>% filter(year == 2019)
```

GDP table

```
str(gdp)
```

```
## tibble [382 x 24] (S3: tbl_df/tbl/data.frame)
##  $ NUTS1 Region: chr [1:382] "North East" "North East" "North East" "North East" ...
##  $ LA code     : chr [1:382] "E06000001" "E06000002" "E06000003" "E06000004" ...
##  $ LA name     : chr [1:382] "Hartlepool" "Middlesbrough" "Redcar and Cleveland" "Stockton-on-Tees"
##  $ 1998        : num [1:382] 1022 1693 1799 2994 1634 ...
##  $ 1999        : num [1:382] 1073 1799 2229 3105 1668 ...
##  $ 2000        : num [1:382] 1081 1828 2131 3170 1796 ...
##  $ 2001        : num [1:382] 1041 1837 2034 3194 1924 ...
##  $ 2002        : num [1:382] 1102 1925 2102 3400 2067 ...
##  $ 2003        : num [1:382] 1160 2007 2256 3681 2184 ...
##  $ 2004        : num [1:382] 1203 2189 2280 3889 2345 ...
##  $ 2005        : num [1:382] 1237 2303 2176 4046 2335 ...
##  $ 2006        : num [1:382] 1299 2466 2200 4358 2387 ...
##  $ 2007        : num [1:382] 1349 2564 2256 4518 2336 ...
##  $ 2008        : num [1:382] 1428 2646 2339 4681 2407 ...
##  $ 2009        : num [1:382] 1403 2666 2316 4845 2297 ...
##  $ 2010        : num [1:382] 1456 2669 2620 4788 2450 ...
##  $ 2011        : num [1:382] 1508 2733 2667 4817 2522 ...
##  $ 2012        : num [1:382] 1496 2842 2792 4732 2547 ...
##  $ 2013        : num [1:382] 1535 2728 2808 4847 2771 ...
##  $ 2014        : num [1:382] 1557 2762 2544 5448 2937 ...
##  $ 2015        : num [1:382] 1636 2894 2061 6121 3090 ...
##  $ 2016        : num [1:382] 1731 3112 2005 6126 2916 ...
##  $ 2017        : num [1:382] 1728 3215 2061 6040 3223 ...
##  $ 20183       : num [1:382] 1732 3388 2159 5885 3076 ...
```

```

sum(is.na(gdp$`2018`)) ## good news
```

No NAs yay

```
## [1] 0
```

```

gdp = gdp %>% rename(Region = `NUTS1 Region`,
                     areaCode = `LA code`,
                     areaName = `LA name`) %>%
  pivot_longer(-c(Region, areaCode, areaName),
               names_to = 'year',
               values_to = 'GDP(millionGBP)') %>%
  mutate(year = as.numeric(year))
```

Rename, pivot, mutate Format year as it kep the superscript from the excel sheet

```
gdp$year = ifelse(gdp$year == 20183, 2018, gdp$year)
```

```

gdp = updateCodes(gdp)

combs_dup = gdp %>%
  group_by(areaCode, areaName, year) %>% summarise(duplicat = n()>1)

print(
  gdp[gdp$areaCode %in%
              combs_dup$areaCode[combs_dup$duplicat==T],c(2,3,4)] %>%
    arrange(areaCode, areaName,year), n = 100)
```

Update codes

```
## # A tibble: 0 x 3
## # ... with 3 variables: areaCode <chr>, areaName <chr>, year <dbl>
```

Many duplicates now

This function combines entries by taking the **sum** when they are duplicated

```
for (row in which(combs_dup$duplicat==T)){
  entry = combs_dup[row,]

  gdp[(gdp$areaCode == entry$areaCode &
             gdp$areaName == entry$areaName &
             gdp$year == entry$year),
           !colnames(gdp) %in% c('areaCode','areaName', 'year')] =
    t(apply(gdp[(gdp$areaCode == entry$areaCode &
                         gdp$areaName == entry$areaName &
                         gdp$year == entry$year),
                     !colnames(gdp) %in% c('areaCode','areaName', 'year')],2,sum))

}

print(
  gdp[gdp$areaCode %in%
        combs_dup$areaCode[combs_dup$duplicat==T],c(2,3,4, 5)] %>%
    arrange(areaCode, areaName,year), n = 100)
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: areaCode <chr>, areaName <chr>, year <dbl>,
## #   `GDP(millionGBP)` <dbl>
```

Check what min and max years are

```
min(gdp$year)
```

```
## [1] 1998
```

```
max(gdp$year)
```

```
## [1] 2018
```

```
gdp_time = gdp
gdp = gdp_time %>% filter(year == max(year))
```

Unemployment

This one is easy to handle as it is almost tidy, we just got to choose the column we are interested in which is the one with the unemployment rate in 2019. Because it is 2019 arleady, we don't even need to update the LAD codes

```
str(unemployment)
```

```
## tibble [372 x 209] (S3: tbl_df/tbl/data.frame)
##  $ UALAD               : chr [1:372] NA NA "County Durham" "Darlington" ...
##  $ ...2                : chr [1:372] NA NA "E06000047" "E06000005" ...
##  $ ...3                : logi [1:372] NA NA NA NA NA NA ...
##  $ 1996/97             : chr [1:372] "Rate" NA ":" "9.3000000000000007" ...
##  $ ...5                : chr [1:372] "+/- (%)2" NA ":" "2" ...
##  $ ...6                : logi [1:372] NA NA NA NA NA NA ...
##  $ 1997/98             : chr [1:372] "Rate" NA ":" "8" ...
##  $ ...8                : chr [1:372] "+/- (%)2" NA ":" "1.8" ...
##  $ ...9                : logi [1:372] NA NA NA NA NA NA ...
##  $ 1998/99             : chr [1:372] "Rate" NA ":" "7" ...
##  $ ...11               : chr [1:372] "+/- (%)2" NA ":" "1.8" ...
##  $ ...12               : logi [1:372] NA NA NA NA NA NA ...
##  $ 1999/2000           : chr [1:372] "Rate" NA ":" "8.0999999999999996" ...
##  $ ...14               : chr [1:372] "+/- (%)2" NA ":" "2" ...
##  $ ...15               : logi [1:372] NA NA NA NA NA NA ...
##  $ 2000/01             : chr [1:372] "Rate" NA ":" "6.9000000000000004" ...
##  $ ...17               : chr [1:372] "+/- (%)2" NA ":" "1.1000000000000001" ...
##  $ ...18               : logi [1:372] NA NA NA NA NA NA ...
##  $ 2001/02             : chr [1:372] "Rate" NA ":" "6.5999999999999996" ...
##  $ ...20               : chr [1:372] "+/- (%)2" NA ":" "1.1000000000000001" ...
##  $ ...21               : logi [1:372] NA NA NA NA NA NA ...
##  $ 2002/03             : chr [1:372] "Rate" NA ":" "5.5999999999999996" ...
##  $ ...23               : chr [1:372] "+/- (%)2" NA ":" "1" ...
##  $ ...24               : logi [1:372] NA NA NA NA NA NA ...
##  $ 2003/04             : chr [1:372] "Rate" NA ":" "5" ...
##  $ ...26               : chr [1:372] "+/- (%)2" NA ":" "0.90000000000000002" ...
##  $ ...27               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jan 2004 to Dec 2004: chr [1:372] "Rate" NA "4.7999999999999998" "4.4000000000000004" ...
##  $ ...29               : chr [1:372] "+/- (%)2" NA ":" "0.80000000000000004" ...
##  $ ...30               : logi [1:372] NA NA NA NA NA NA ...
##  $ Apr 2004 to Mar 2005: chr [1:372] "Rate" NA "4.7999999999999998" "4.5" ...
##  $ ...32               : chr [1:372] "+/- (%)2" NA ":" "0.90000000000000002" ...
##  $ ...33               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jul 2004 to Jun 2005: chr [1:372] "Rate" NA "4.9000000000000004" "4.5999999999999996" ...
##  $ ...35               : chr [1:372] "+/- (%)2" NA ":" "0.90000000000000002" ...
##  $ ...36               : logi [1:372] NA NA NA NA NA NA ...
##  $ Oct 2004 to Sep 2005: chr [1:372] "Rate" NA "4.7999999999999998" "4.5999999999999996" ...
##  $ ...38               : chr [1:372] "+/- (%)2" NA ":" "0.90000000000000002" ...
##  $ ...39               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jan 2005 to Dec 2005: chr [1:372] "Rate" NA "4.7000000000000002" "4.7999999999999998" ...
##  $ ...41               : chr [1:372] "+/- (%)2" NA ":" "0.90000000000000002" ...
##  $ ...42               : logi [1:372] NA NA NA NA NA NA ...
##  $ Apr 2005 to Mar 2006: chr [1:372] "Rate" NA "5.5" "4.90961113" ...
##  $ ...44               : chr [1:372] "+/- (%)2" NA ":" "1.0438646008000001" ...
##  $ ...45               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jul 2005 to Jun 2006: chr [1:372] "Rate" NA "5.5" "4.7195016299999999" ...
##  $ ...47               : chr [1:372] "+/- (%)2" NA ":" "1.0602798555999999" ...
##  $ ...48               : logi [1:372] NA NA NA NA NA NA ...
##  $ Oct 2005 to Sep 2006: chr [1:372] "Rate" NA "5.7999999999999998" "4.7410787299999999" ...
##  $ ...50               : chr [1:372] "+/- (%)2" NA ":" "1.0497842712000001" ...
##  $ ...51               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jan 2006 to Dec 2006: chr [1:372] "Rate (%)" NA "5.9000000000000004" "5.1044561899999996" ...
##  $ ...53               : chr [1:372] "+/- (%)2" NA ":" "1.0330758592" ...
##  $ ...54               : logi [1:372] NA NA NA NA NA NA ...
##  $ Apr 2006 to Mar 2007: chr [1:372] "Rate (%)" NA "5.5999999999999996" "5.1906668099999997" ...
##  $ ...56               : chr [1:372] "+/- (%)2" NA ":" "0.96572381080000003" ...
##  $ ...57               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jul 2006 to Jun 2007: chr [1:372] "Rate (%)" NA "5.4000000000000004" "5.5070090500000006" ...
##  $ ...59               : chr [1:372] "+/- (%)2" NA ":" "0.88716959799999995" ...
##  $ ...60               : logi [1:372] NA NA NA NA NA NA ...
##  $ Oct 2006 to Sep 2007: chr [1:372] "Rate (%)" NA "5.2999999999999998" "5.8317064100000007" ...
##  $ ...62               : chr [1:372] "+/- (%)2" NA ":" "0.96383774200000005" ...
##  $ ...63               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jan 2007 to Dec 2007: chr [1:372] "Rate (%)" NA "5.0999999999999996" "5.7912505000000003" ...
##  $ ...65               : chr [1:372] "+/- (%)2" NA ":" "1.0579039240000001" ...
##  $ ...66               : logi [1:372] NA NA NA NA NA NA ...
##  $ Apr 2007 to Mar 2008: chr [1:372] "Rate (%)" NA "5.2000000000000002" "5.9475737200000003" ...
##  $ ...68               : chr [1:372] "+/- (%)2" NA ":" "1.0874065888" ...
##  $ ...69               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jul 2007 to Jun 2008: chr [1:372] "Rate (%)" NA "5.5" "5.7331496" ...
##  $ ...71               : chr [1:372] "+/- (%)2" NA ":" "1.1056296692000001" ...
##  $ ...72               : logi [1:372] NA NA NA NA NA NA ...
##  $ Oct 2007 to Sep 2008: chr [1:372] "Rate (%)" NA "5.5999999999999996" "5.8950876999999995" ...
##  $ ...74               : chr [1:372] "+/- (%)2" NA ":" "1.0622337992000002" ...
##  $ ...75               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jan 2008 to Dec 2008: chr [1:372] "Rate (%)" NA "6.2999999999999998" "6.3165477799999996" ...
##  $ ...77               : chr [1:372] "+/- (%)2" NA ":" "1.1078058572" ...
##  $ ...78               : logi [1:372] NA NA NA NA NA NA ...
##  $ Apr 2008 to Mar 2009: chr [1:372] "Rate (%)" NA "7" "6.9856878900000003" ...
##  $ ...80               : chr [1:372] "+/- (%)2" NA ":" "1.1598450136" ...
##  $ ...81               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jul 2008 to Jun 2009: chr [1:372] "Rate (%)" NA "7.7999999999999998" "7.8599533799999994" ...
##  $ ...83               : chr [1:372] "+/- (%)2" NA ":" "1.2499008984" ...
##  $ ...84               : logi [1:372] NA NA NA NA NA NA ...
##  $ Oct 2008 to Sep 2009: chr [1:372] "Rate (%)" NA "8.1999999999999993" "8.3100855300000003" ...
##  $ ...86               : chr [1:372] "+/- (%)2" NA ":" "1.269439472" ...
##  $ ...87               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jan 2009 to Dec 2009: chr [1:372] "Rate (%)" NA "8.5999999999999996" "8.6434309099999993" ...
##  $ ...89               : chr [1:372] "+/- (%)2" NA ":" "1.2252655800000001" ...
##  $ ...90               : logi [1:372] NA NA NA NA NA NA ...
##  $ Apr 2009 to Mar 2010: chr [1:372] "Rate (%)" NA "8.9000000000000004" "8.9020545699999989" ...
##  $ ...92               : chr [1:372] "+/- (%)2" NA ":" "1.2535635112000001" ...
##  $ ...93               : logi [1:372] NA NA NA NA NA NA ...
##  $ Jul 2009 to Jun 2010: chr [1:372] "Rate (%)" NA "8.8000000000000007" "8.9156750099999993" ...
##  $ ...95               : chr [1:372] "+/- (%)2" NA ":" "1.2760016303999999" ...
##  $ ...96               : logi [1:372] NA NA NA NA NA NA ...
##  $ Oct 2009 to Sep 2010: chr [1:372] "Rate (%)" NA "8.5" "8.508013909999999" ...
##  $ ...98               : chr [1:372] "+/- (%)2" NA ":" "1.2405666531999999" ...
##  $ ...99               : logi [1:372] NA NA NA NA NA NA ...
##   [list output truncated]
```

```
unemployment = unemployment[3:nrow(unemployment), -seq(3, ncol(unemployment), 3)]
```

The date format for this table is confusing so I am not gonna take care of the tidying yet

```
unemployment_time = unemployment[, c(1,2,seq(3, ncol(unemployment), 2))]
colnames(unemployment_time)[1:2] = c('areaName', 'areaCode')
unemployment_time = unemployment_time %>% pivot_longer(-c(areaName, areaCode),
                                                       names_to = 'year',
                                                       values_to = 'UnempRate')

unemployment = unemployment[, c(1, 2, ncol(unemployment)-1)]

colnames(unemployment) = c('areaName', 'areaCode', 'UnempRate')
unemployment$year = 2019
```

Merge Data

For each dataset the data that has been taken it's that for the latest available year, we'll log what that is for each dataframe

```
ONS_time = Reduce(function(x,y) merge(x = x, y = y,
                                      by = c('areaCode', 'areaName', 'year'),
                                      all = TRUE),
                  list(popclean_time, gdp_time, lifeexpect_time, suicides_time,
                       unemployment, wellbeing_time, work_time))

write.csv(ONS_time, 'CleanData/ONStime.csv')
## unemployment_time year format is impossible to understand and it is too much time to preprocess it
```

Time-series data frame

```
years_for_dfs = data.frame(df = character(),
                           year = integer(),
                           yearColumn = character())


for (obj in ls()){
  if ('data.frame' %in% class(eval(parse(text = obj))) &
      (obj != 'years_for_dfs' &
      obj != 'tst' &
      !grepl('_time', obj) &
      obj != 'unemployment') &
      obj != 'ONS_time'
      ){
    years_for_dfs[nrow(years_for_dfs)+1, 'df'] = obj
    curr_df = eval(parse(text = obj))
    years_for_dfs[nrow(years_for_dfs), 'year'] =
      ifelse('year' %in% colnames(curr_df),
             curr_df$year, NA)
    years_for_dfs[nrow(years_for_dfs), 'yearColumn'] =
      ifelse('year' %in% colnames(curr_df),
             'year', NA)
```

```r
    #remove year column
    if ('year' %in% colnames(curr_df)){
      curr_df = curr_df[,-which(colnames(curr_df) == 'year')]
    }

    eval(parse(text = paste(obj, ' = curr_df')))
  }
}
kable(years_for_dfs)
```

Most current data

df	year	yearColumn
age	2018	year
agetotal	2018	year
combs_dup	1998	year
entry	2018	year
gdp	2018	year
lifeexpect	2017	year
popclean	2018	year
population	2005	year
suicides	2018	year
wellbeing	2019	year
work	2019	year
work_ts	2018	year

```r
remove(obj)
```

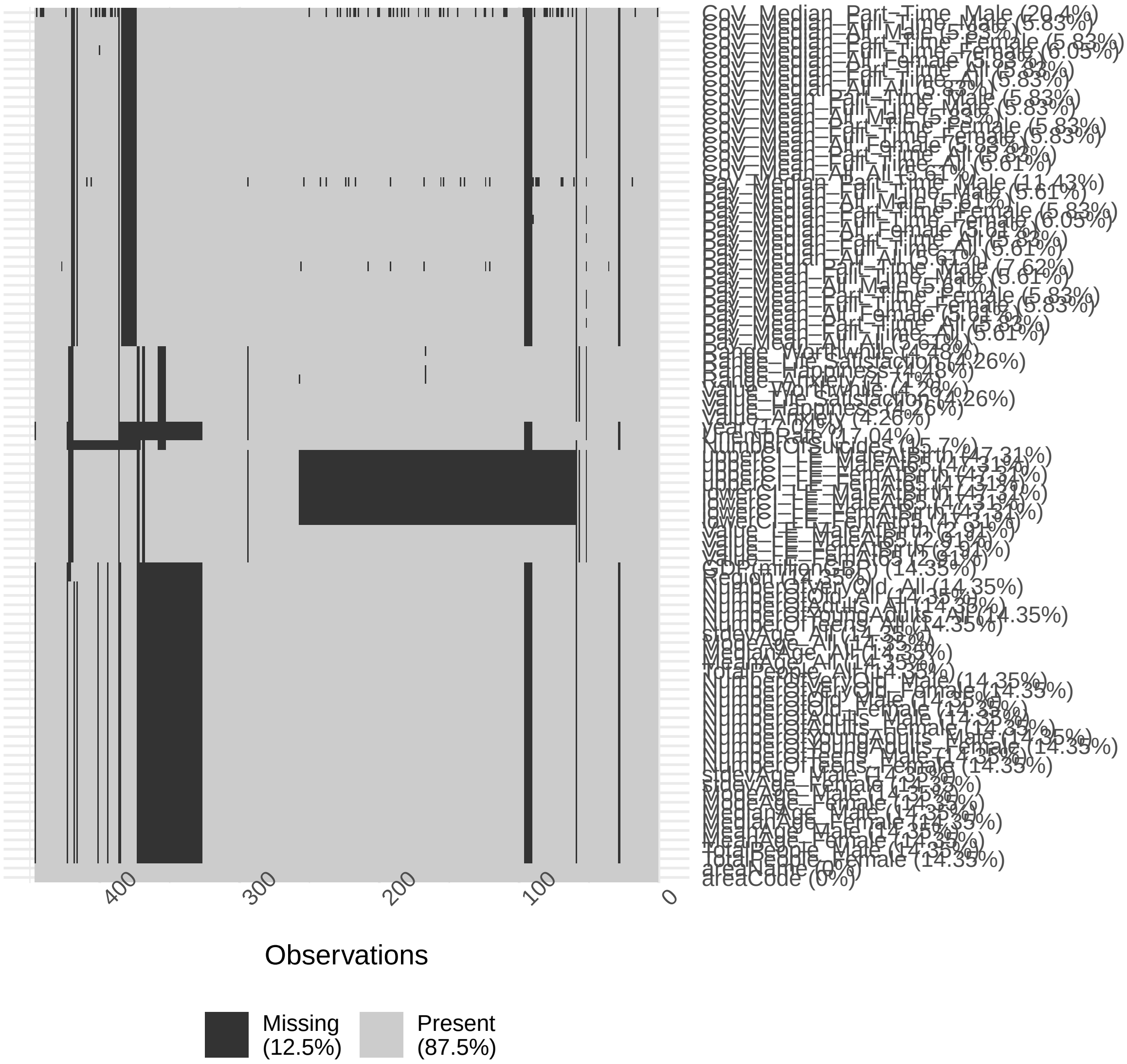
MERGE!

As it can be seen in the function call below we are performing outer joins(`all = TRUE`)

```r
ONS = Reduce(function(x,y) merge(x = x, y = y,
                                 by = c('areaCode', 'areaName'),
                                 all = TRUE),
             list(popclean, gdp, lifeexpect, suicides,
                  unemployment, wellbeing, work))
```

Finally we can visualise the missingness in the final dataset, though you may need a magnifier!

```r
vis_miss(ONS)+coord_flip()
```



Save all data

```r
write.csv(ONS, 'CleanData/ONS.csv')
```

after this run:

```r
rmarkdown::render('01_tidyingOriginalData.R',
                  output_format = c('html_document','pdf_document'))
```

Appendix B

Results appendix

B.1 UMAP PAN embedding calibration

Plot available in figure B.1.

B.2 Alternative clustering methods

Figures B.2, B.3, B.4, B.5 show the result of applying t-SNE, DBSCAN, k-means and Gaussian mixture models after the LAD data has been reduced to 20 PCA components (keeping 90% of variance).

B.3 PCA of the whole dataset + external labelling by region

PCA scree-plot

A scree plot with the % variance per principal component can be seen in figure B.6

PCA embedding coloured by region

Clear separation of a London ellipse can be seen in figure B.7

B.4 Univariate analysis

Volcano plots from univariate linear models assessing the effect of individual predictor variables on the different life expectancy variables in figure B.8

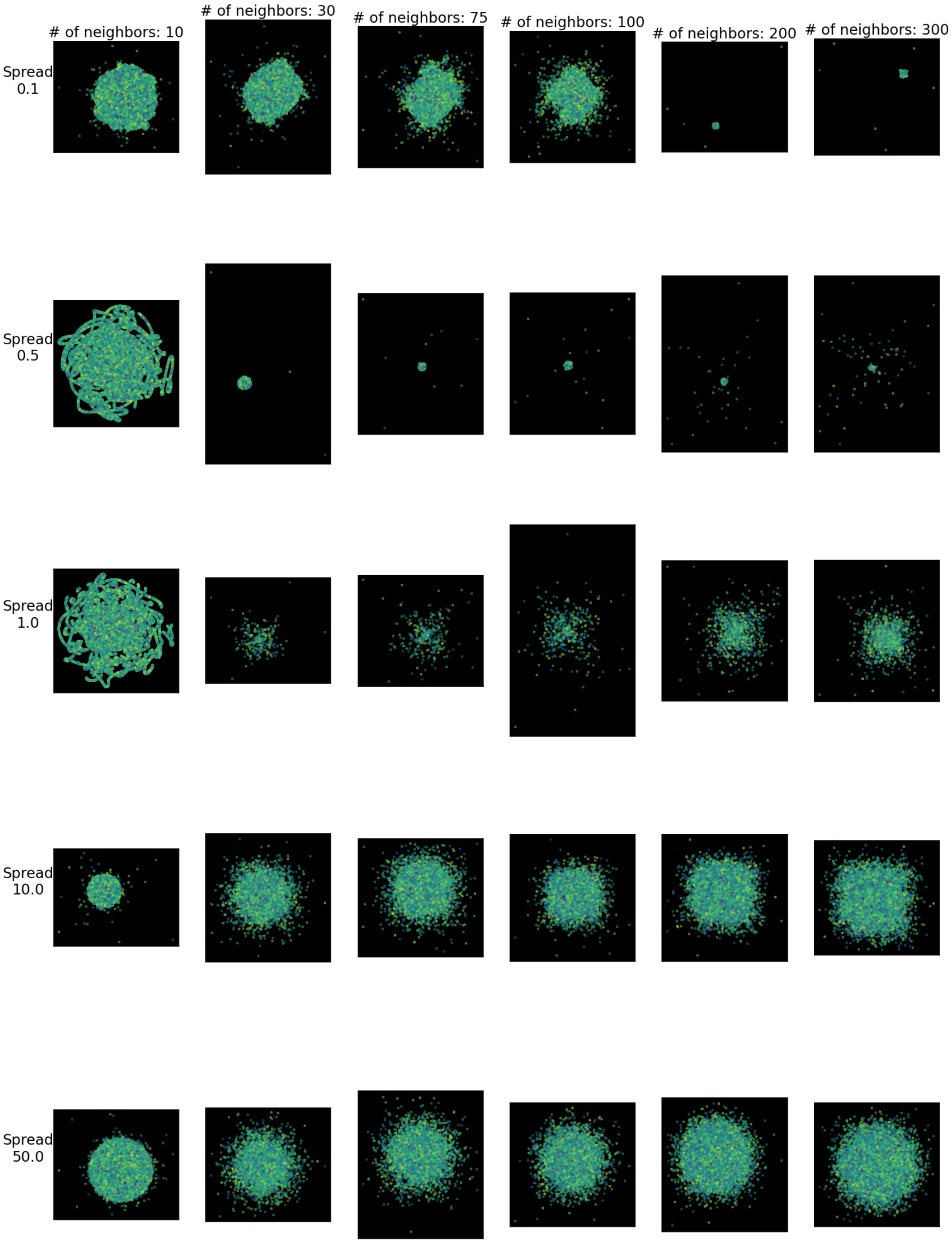


Figure B.1: UMAP embedding of 100,000 PANs with different hyper-parameters. The points are colored by the LE at birth value

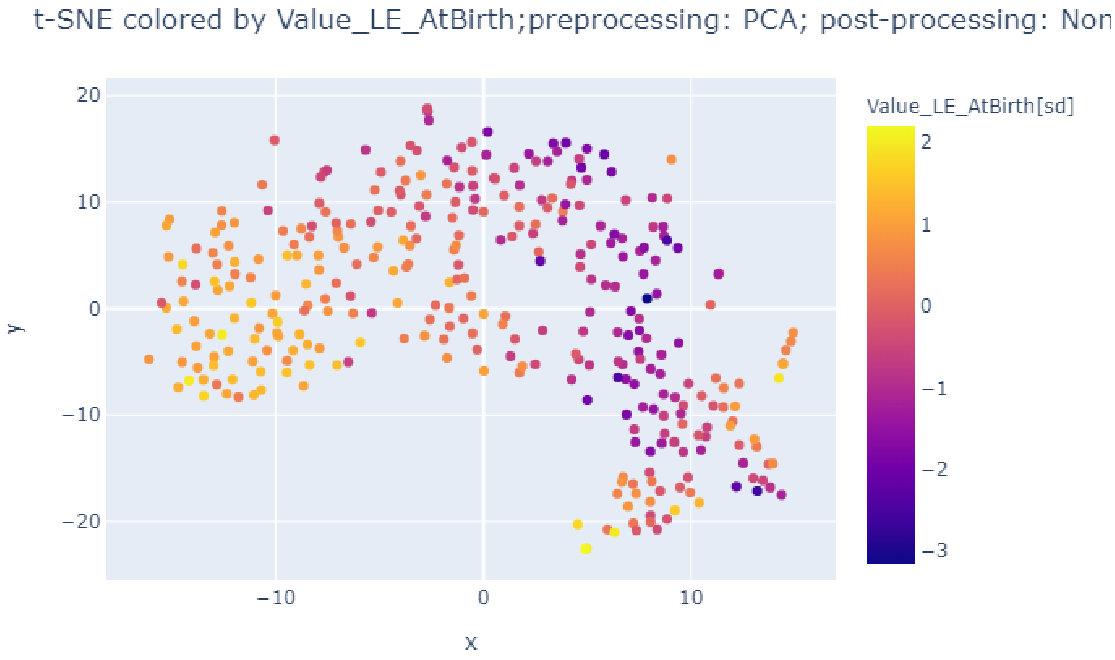


Figure B.2: PCA + t-SNE applied to LAD data

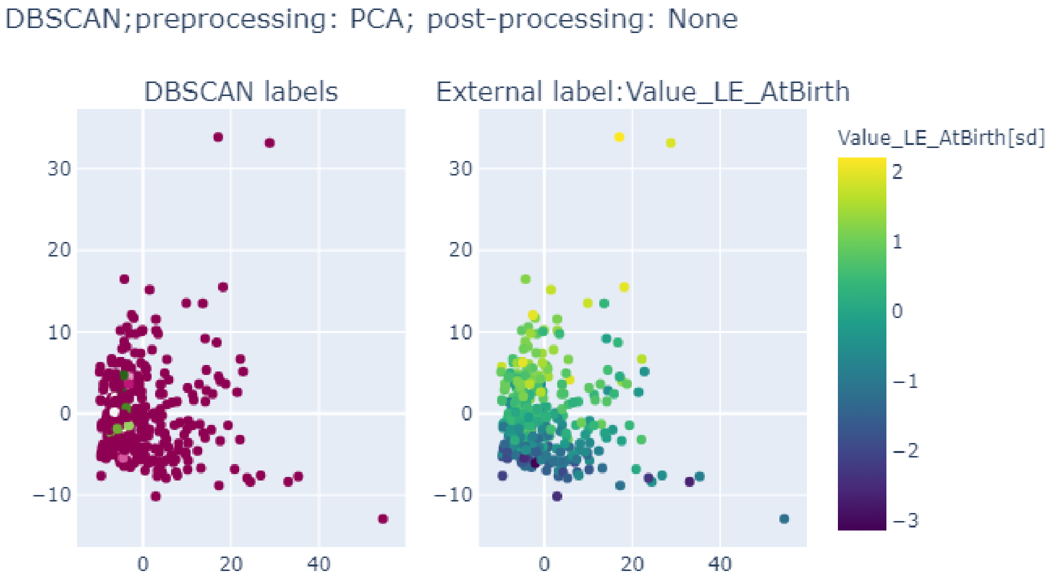


Figure B.3: PCA + DBSCAN applied to LAD data

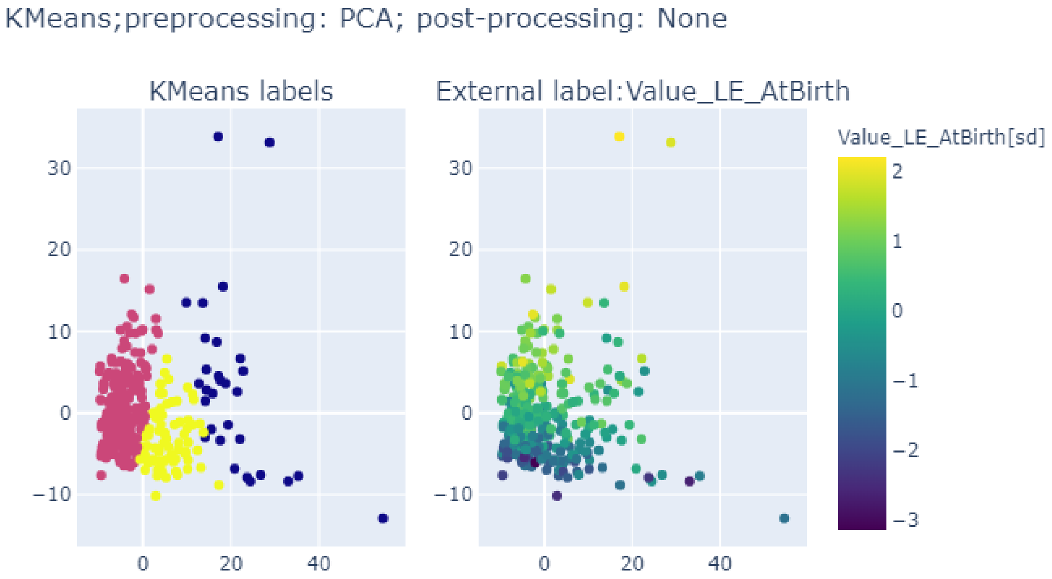


Figure B.4: PCA + K-means applied to LAD data

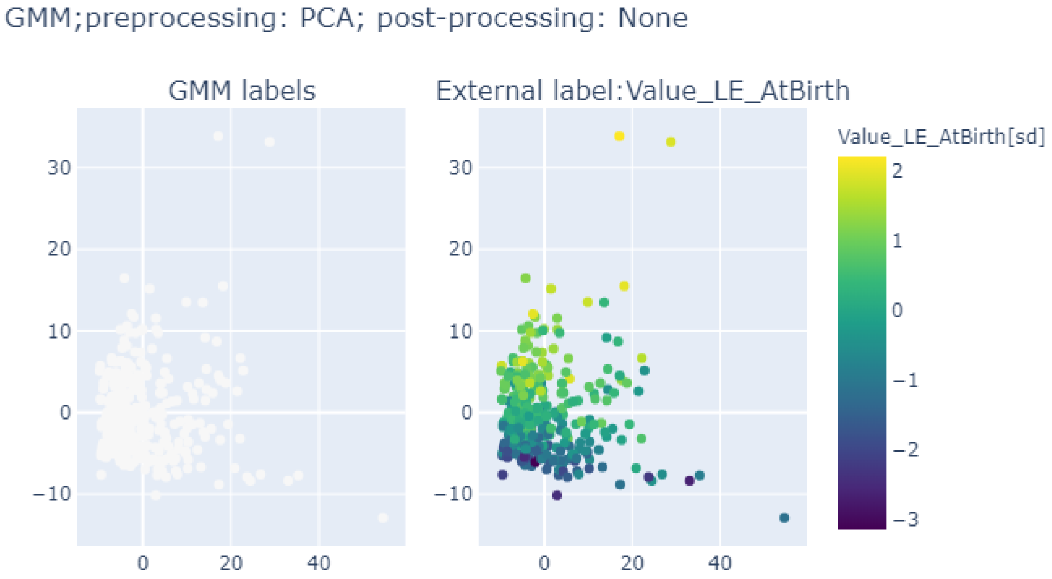


Figure B.5: PCA + Gaussian mixture models applied to LAD data

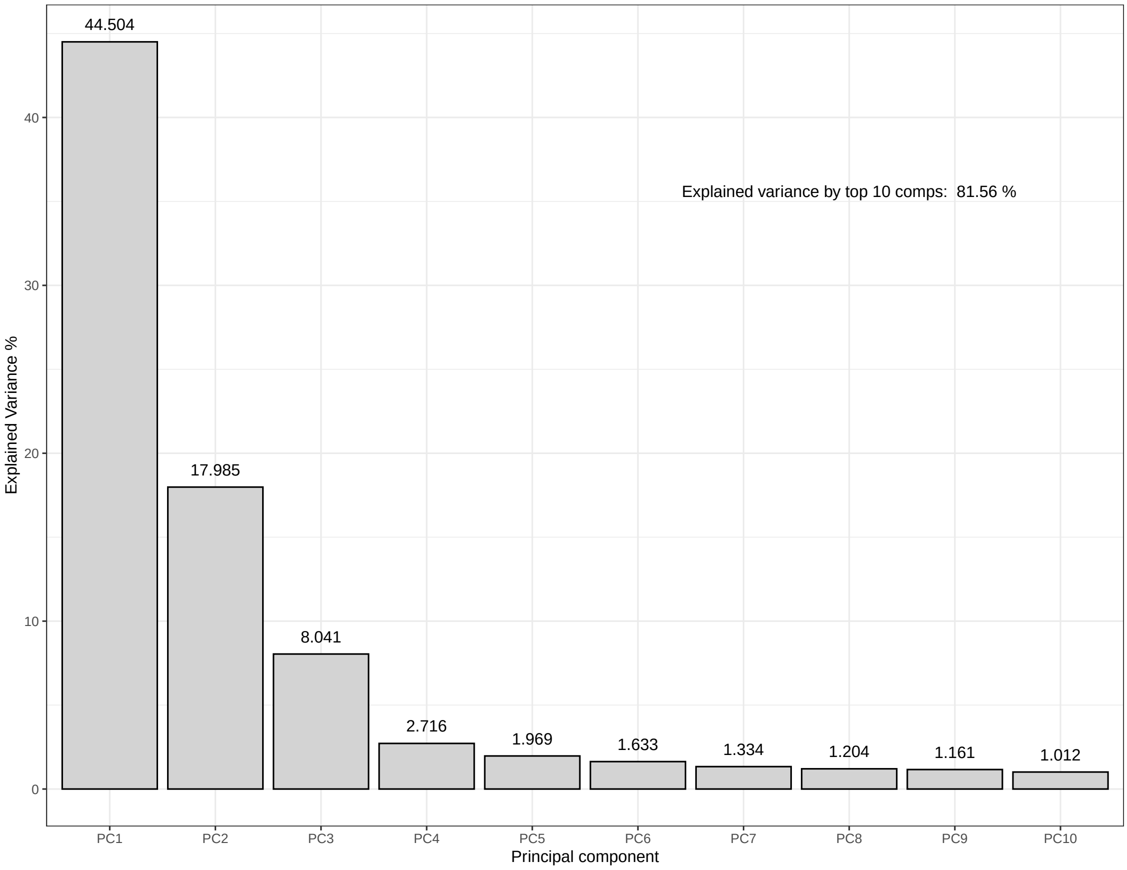


Figure B.6: Scree plot of Visa+ONS LAD data PCA

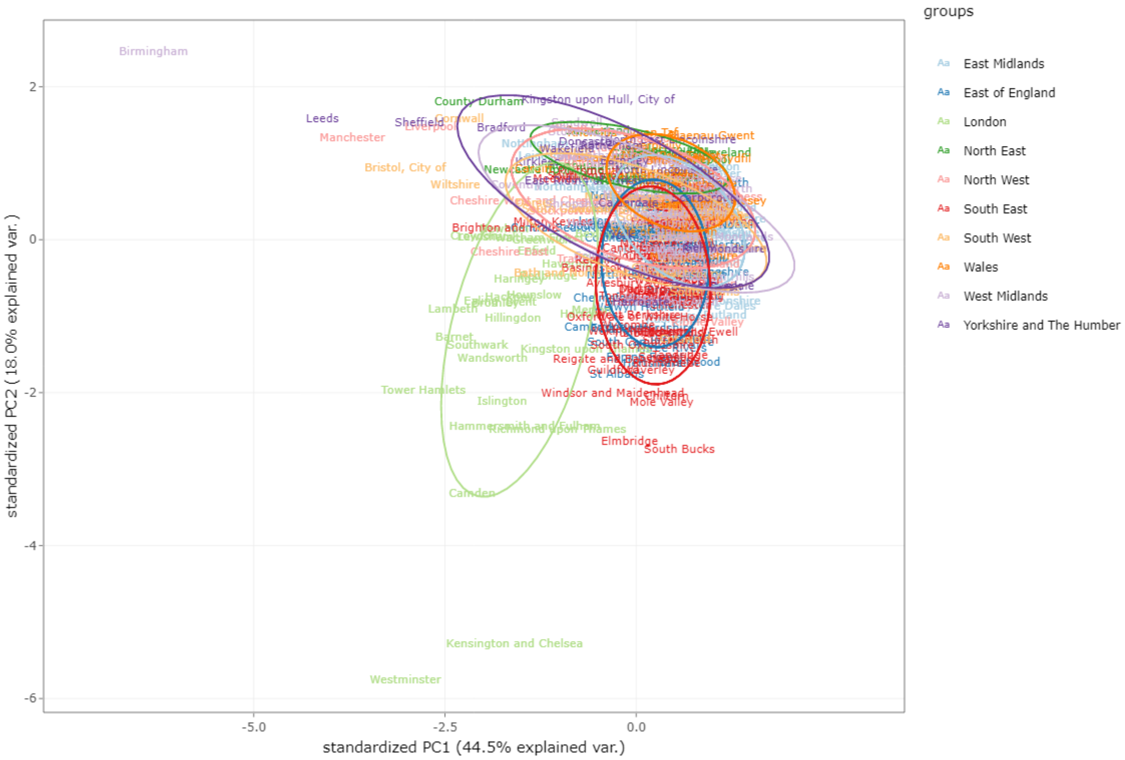


Figure B.7: PCA embedding coloured by region

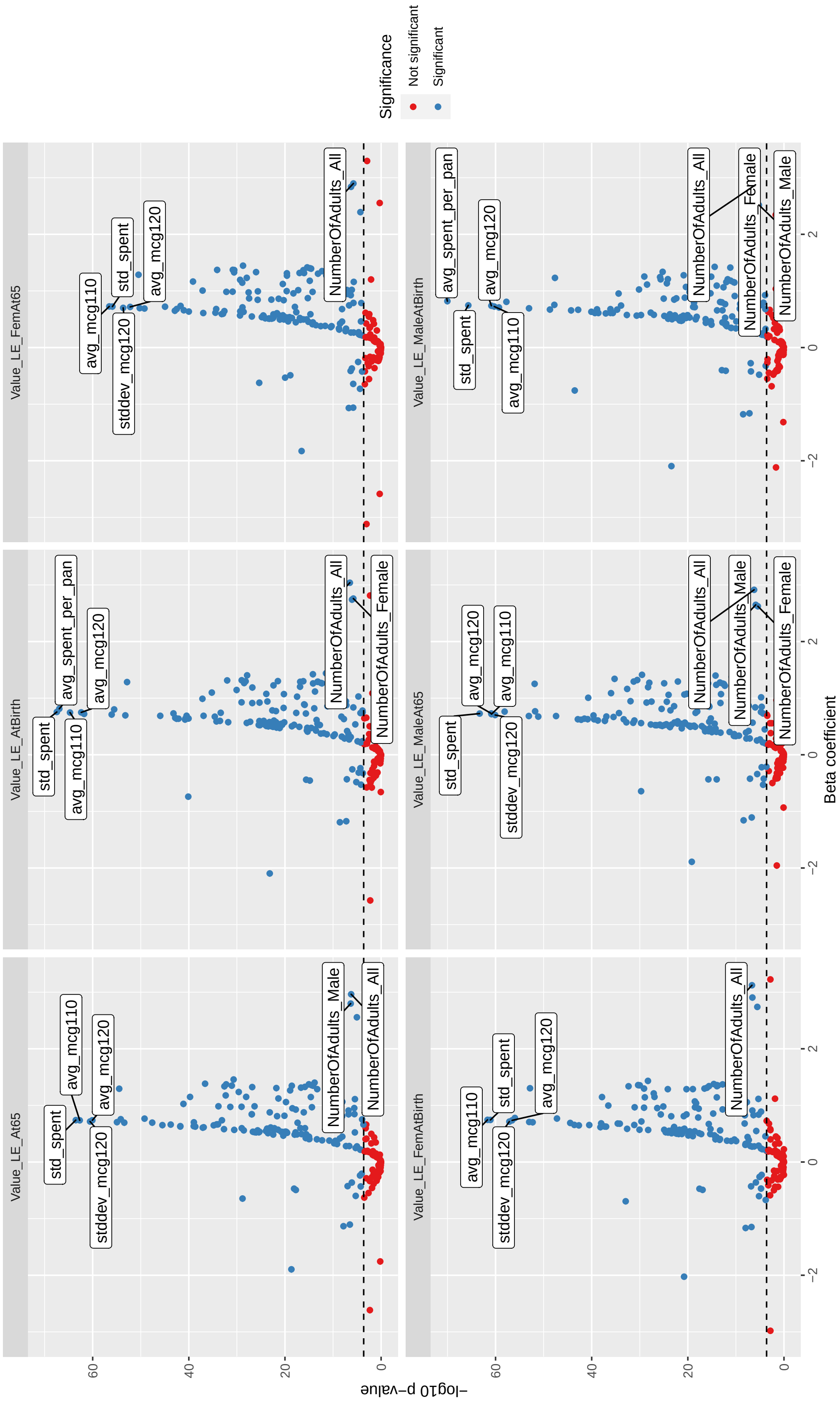


Figure B.8: Volcano plot for different LE variables

Bibliography

[1] Dr Pilar, Garcia Dr, and Mark Mccarthy. *Measuring Health A STEP IN THE DEVELOPMENT OF CITY HEALTH PROFILES*. Tech. rep.

[2] World Health Organization. *CONSTITUTION OF THE WORLD HEALTH ORGANIZATION*. Tech. rep. World Health Organisation.

[3] *WEALTH | meaning in the Cambridge English Dictionary*. URL: https://dictionary.cambridge.org/dictionary/english/wealth.

[4] Michael Marmot. "Connection between wealth and health?" In: *The Lancet* 382.9905 (2013), pp. 1623–1624. ISSN: 01406736. DOI: 10.1016/s0140-6736(13)62355-7. URL: www.thelancet.com.

[5] John A Sturgeon et al. "The psychosocial context of financial stress: Implications for inflammation and psychological health". In: *Psychosomatic Medicine* 78.2 (2016), pp. 134–143. ISSN: 15347796. DOI: 10.1097/PSY.0000000000000276.

[6] Karen Rowlingson. *Does income inequality cause health and social problems? This report provides an independent review of the evidence about the impact of inequality*. Tech. rep. 2011. URL: www.jrf.org.uk.

[7] *Fair Society, Healthy Lives The Marmot Review*. Tech. rep.

[8] Ali H Mokdad et al. *Actual Causes of Death in the United States, 2000*. Tech. rep. URL: www.jama.com.

[9] Earl S. Ford et al. "Healthy lifestyle behaviors and all-cause mortality among adults in the United States". In: *Preventive Medicine* 55.1 (July 2012), pp. 23–27. ISSN: 00917435. DOI: 10.1016/j.ypmed.2012.04.016. URL: /pmc/articles/PMC4688898/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4688898/.

[10] Harvey Checkoway, Neil Pearce, and David Kriebel. *Selecting appropriate study designs to address specific research questions in occupational epidemiology*. Sept. 2007. DOI: 10.1136/oem.2006.029967. URL: /pmc/articles/PMC2092571/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2092571/.

[11] Ali H. Mokdad and Patrick L. Remington. "Measuring health behaviors in populations". In: *Preventing Chronic Disease* 7.4 (2010). ISSN: 21665435. URL: http://www.cdc.gov/pcd/issues/2010/jul/10_0010.htm.Accessed[date]..

[12] Ali H Mokdad. "The Behavioral Risk Factors Surveillance System: Past, Present, and Future". In: (2009). DOI: 10.1146/annurev.publhealth.031308.100226. URL: www.annualreviews.org.

[13] Graham Kalton. *Introduction to Survey Sampling*. Ed. by Michael S. Lewis-Beck. SAGE University Papers, 1983. URL: https://eclass.uoa.gr/modules/document/file.php/ECON261/SAmpling%20Surveys/Graham%20Kalton%20Introduction%20to%20Survey%20Sampling%20Quantitative%20Applications%20in%20the%20Social%20Sciences%201983.pdf.

[14] James J. Heckman. "Sample selection bias as a specification error". In: *Applied Econometrics* 31.3 (2013), pp. 129–137. ISSN: 24106445. DOI: 10.2307/1912352.

[15] *Recall bias - Catalog of Bias*. URL: https://catalogofbias.org/biases/recall-bias/.

[16] Paul Lavrakas. "Social Desirability". In: *Encyclopedia of Survey Research Methods*. Sage Publications, Inc., Apr. 2013. DOI: 10.4135/9781412963947.n537.

[17] *Reporting Biases - Catalog of Bias*. URL: https://catalogofbias.org/biases/reporting-biases/.

[18] Anya Skatova, Kate Shiells, and Andy Boyd. "Attitudes towards transactional data donation and linkage in a longitudinal population study: evidence from the Avon Longitudinal Study of Parents and Children". In: *Wellcome Open Research* 4 (2019), p. 192. ISSN: 2398-502X. DOI: 10.12688/wellcomeopenres.15557.1. URL: https://doi.org/10.12688/wellcomeopenres.15557.1.

[19] *Donating personal transactional data for research | The Alan Turing Institute*. URL: https://www.turing.ac.uk/research/research-projects/donating-personal-transactional-data-research.

[20] Visa School of Public Policy. *How Electronic Payments Work - YouTube*. 2017. URL: https://www.youtube.com/watch?v=s5HrEyEStYE.

[21] *VisaNet | Electronic Payments Network | Visa*. URL: https://www.visa.co.uk/about-visa/visanet.html.

[22] Visa School of Public Policy. *Visa School of Public Policy (VSPP) - YouTube Channel*. URL: https://www.youtube.com/channel/UCnAWxrciYGp51p5rdMx-NWQ.

[23] Visa Inc. *VisaCommunication - YouTube*. URL: https://www.youtube.com/channel/UCKAhE4PZPQMfyXbsi58evHA.

[24] Visa Inc. *Visa Fiscal Annual Report 2019*. Tech. rep. 2019. URL: https://s24.q4cdn.com/307498497/files/doc_downloads/Visa_Inc_Fiscal_2019_Annual_Report.pdf.

[25] Global Data. *UK Cards & Payments: Opportunities and Risks to 2023*. URL: https://www.researchandmarkets.com/reports/4987487/uk-cards-and-payments-opportunities-and-risks-to.

[26] Kate Ann Levin. "Study Design VI-Ecological Studies". In: *Evidence-Based Dentistry* 7 (2003), pp. 60–61. DOI: 10.1038/sj.ebd.6400454. URL: www.nature.com/ebd.

[27] Office for National Statistics. *Health state life expectancy at birth and at age 65 by local areas, UK - Office for National Statistics*. 2018. URL: https://www.ons.gov.uk/datasets/life-expectancy-local-authority/editions/time-series/versions/1.

[28] Office for National Statistics. *Health state life expectancies, UK QMI - Office for National Statistics*. 2018. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/healthandsocialcare/healthandlifeexpectancies/methodologies/healthstatelifeexpectanciesukqmi.

[29] Office for National Statistics. *Suicide rates in the UK QMI - Office for National Statistics*. 2019. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/deaths/methodologies/suicideratesintheukqmi.

[30] *Suicide registrations in England and Wales by local authority - Office for National Statistics*. URL: https://www.ons.gov.uk/datasets/suicides-in-the-uk/editions/time-series/versions/1.

[31] *Suicide registrations in England and Wales by local authority - Office for National Statistics*. URL: https://www.ons.gov.uk/datasets/suicides-in-the-uk/editions/2018/versions/1.

[32] *Personal well-being in the UK QMI - Office for National Statistics*. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/wellbeing/methodologies/personalwellbeingintheukqmi.

[33] *Personal well-being estimates by local authority - Office for National Statistics*. URL: https://www.ons.gov.uk/datasets/wellbeing-local-authority/editions/time-series/versions/2.

[34] *The campaign to reach the whole population - Office for National Statistics*. URL: https://www.ons.gov.uk/census/2011census/howourcensusworks/howwetookthe2011census/howwecollectedtheinformation/thecampaigntoreachthewholepopulation.

[35] *Mid-year population estimates QMI - Office for National Statistics*. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates/methodologies/midyearpopulationestimatesqmi.

[36] Vu Quang Viet. *GDP by production approach: A general introduction with emphasis on an integrated economic data collection framework*. Tech. rep. 2009.

[37] *Gross domestic product (GDP) QMI - Office for National Statistics*. URL: https://www.ons.gov.uk/economy/grossdomesticproductgdp/methodologies/grossdomesticproductgdpqmi.

[38] *Regional economic activity by gross domestic product, UK - Office for National Statistics*. URL: https://www.ons.gov.uk/economy/grossdomesticproductgdp/bulletins/regionaleconomicactivitybygrossdomesticproductuk/1998to2018.

[39] *Regional gross value added (balanced) QMI - Office for National Statistics*. URL: https://www.ons.gov.uk/economy/grossvalueaddedgva/methodologies/regionalgrossvalueaddedbalancedqmi#methods.

[40] *A guide to labour market statistics - Office for National Statistics*. URL: https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/employmentandemployeetypes/methodologies/aguidetolabourmarketstatistics#unemployment.

[41] *Labour market in the regions of the UK - Office for National Statistics*. URL: https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/employmentandemployeetypes/bulletins/regionallabourmarket/latest.

[42] *Annual Survey of Hours and Earnings, Low pay and Annual Survey of Hours and Earnings pension results QMI - Office for National Statistics*. URL: https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/earningsandworkinghours/methodologies/annualsurveyofhoursandearningslowpayandannualsurveyofhoursandearningspensionresultsqmi.

[43] *Earnings and hours worked, place of work and residence by local authority: ASHE Tables 7 and 8 - Office for National Statistics*. URL: https://www.ons.gov.uk/datasets/ashe-tables-7-and-8/editions/2019/versions/1.

[44] *National Statistics Postcode Lookup (November 2019) | Open Geography portal*. URL: https://geoportal.statistics.gov.uk/datasets/national-statistics-postcode-lookup-november-2019.

[45] *Postal geography - Office for National Statistics*. URL: https://www.ons.gov.uk/methodology/geography/ukgeographies/postalgeography.

[46] Michael Mayer. *Using missRanger*. 2019. URL: https://cran.r-project.org/web/packages/missRanger/vignettes/vignette_missRanger.html.

[47] Daniel J. Stekhoven and Peter Bühlmann. "Missforest-Non-parametric missing value imputation for mixed-type data". In: *Bioinformatics* 28.1 (Jan. 2012), pp. 112–118. ISSN: 13674803. DOI: 10.1093/bioinformatics/btr597. URL: http://stat.ethz.ch/CRAN/..

[48] Marvin N Wright and Andreas Ziegler. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R". In: *JSS Journal of Statistical Software* 77 (2017). DOI: 10.18637/jss.v077.i01. URL: http://dx.doi.org/10.18637/jss.v077.i01..

[49] Runmin Wei et al. "Missing Value Imputation Approach for Mass Spectrometry-based Metabolomics Data". In: *Scientific Reports* 8.1 (Dec. 2018), pp. 1–10. ISSN: 20452322. DOI: 10.1038/s41598-017-19120-0.

[50] Robert Tibshirani. "Regression Shriknage and Selectino via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 00359246. DOI: 10.2307/2346178. URL: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.7574%20https://www.jstor.org/stable/2346178.

[51] Laurens Van Der Maaten and Geoffrey Hinton. *Visualizing Data using t-SNE*. Tech. rep. 2008, pp. 2579–2605.

[52] Martin Ester et al. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Tech. rep. 1996. URL: www.aaai.org.

[53] Leland McInnes, John Healy, and James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: (Feb. 2018). URL: http://arxiv.org/abs/1802.03426.

[54] *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction — umap 0.4 documentation*. URL: https://umap-learn.readthedocs.io/en/latest/.

[55] Sebastian Raschka, Joshua Patterson, and Corey Nolet. *Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence*. Tech. rep. URL: http://packages.pypy.org.

[56] *Understanding UMAP*. URL: https://pair-code.github.io/understanding-umap/.

[57] *How UMAP Works — umap 0.4 documentation*. URL: https://umap-learn.readthedocs.io/en/latest/how_umap_works.html.

[58] *Births and death by Lower layer Super Output Area (LSOA), England and Wales, 2016 to 2017 - Office for National Statistics*. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/deaths/adhocs/009298birthsanddeathbylowerlayersuperoutputarealsoaeng

[59] *Lower layer Super Output Area population estimates (supporting information) - Office for National Statistics*. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates/datasets/lowersuperoutputareamidyearpopulationesti

[60] Communities Local Government Ministry of Housing. *The English Indices of Deprivation 2019 (IoD2019)*. URL: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/833959/IoD2019_Infographic.pdf.

[61] John Bagnall et al. “Consumer cash usage: a cross-country comparison with payment diary survey data”. In: 2014. ISBN: 978-92-899-1093-4. URL: http://www.ecb.europa.euhttp://ssrn.com/http://www.ecb.europa.eu/pub/scientific/wps/date/html/index.en.htmlhttp://www.ecb.europa.eu/events/conferences/html/131021_ecb_bdf.en.html..

[62] Scott Schuh and Joanna Stavins. “How Consumers Pay: Adoption and Use of Payments”. In: *Accounting and Finance Research* 2.2 (Mar. 2013). ISSN: 1927-5986. DOI: 10.5430/afr.v2n2p1.

[63] *Consumer sentiment and behavior continue to reflect the uncertainty of the COVID-19 crisis | McKinsey*. URL: https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/a-global-view-of-how-consumer-behavior-is-changing-amid-covid-19.

[64] • *Coronavirus: expected changes to consumer spending by product category U.S. 2020 | Statista*. URL: https://www.statista.com/statistics/1105623/coronavirus-expected-changes-to-consumer-spending-by-product-category-us/.

[65] *English Indices of Deprivation 2019 FAQs*. Tech. rep.

[66] Tian Chen et al. “Rank regression: an alternative regression approach for data with outliers”. In: *Shanghai Archives of Psychiatry* 26.5 (Oct. 2014), pp. 310–316. ISSN: 10020829. DOI: 10.11919/j.issn.1002-0829.214148. URL: /pmc/articles/PMC4248265/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4248265/.

[67] Karl Pearson. “ LIII. On lines and planes of closest fit to systems of points in space ”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (Nov. 1901), pp. 559–572. ISSN: 1941-5982. DOI: 10.1080/14786440109462720. URL: https://www.tandfonline.com/doi/abs/10.1080/14786440109462720.

[68] *6.5.14. Algorithms to calculate (build) PCA models — Process Improvement using Data*. URL: https://learnche.org/pid/latent-variable-modelling/principal-component-analysis/algorithms-to-calculate-build-pca-models.

[69] Carnegie Mellon University. “Principal Component Analysis - Advanced Data Analysis”. In: Carnegie Mellon University. URL: https://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch18.pdf.