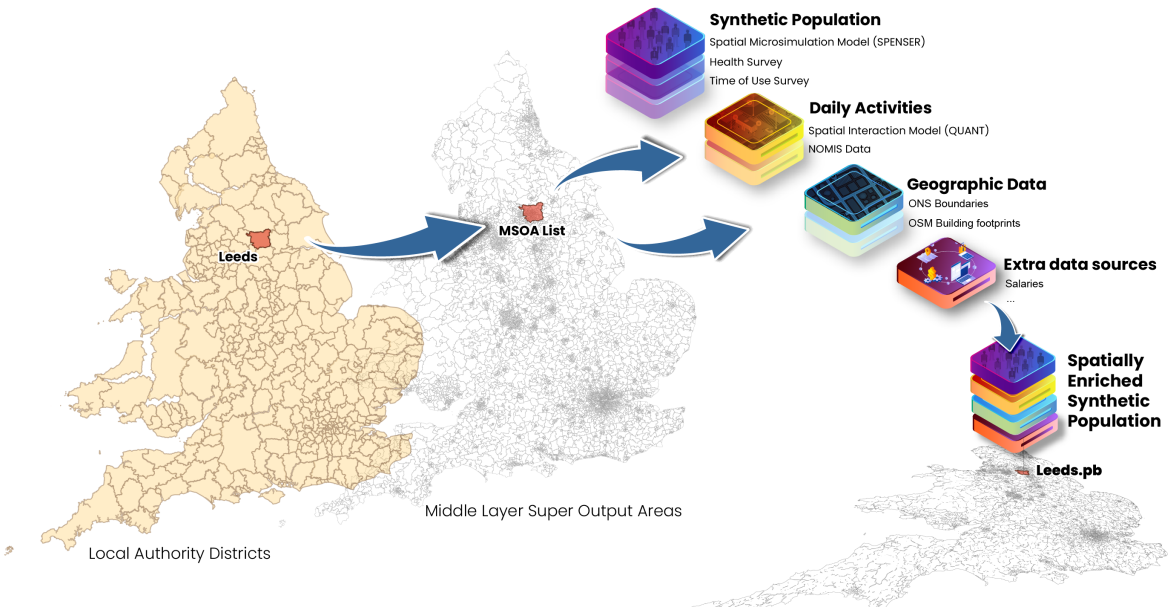
Synthetic Population Catalyst

Table of contents

Introduction



The Synthetic Population Catalyst (SPC) makes it easier for researchers to work with synthetic population data in Great Britain. It combines a variety of data sources and outputs a single file in protocol buffer format, describing the population and its activities in a given study area. The data include socio-demographic, health, salary and daily activity data per person, and information about the venues where people conduct those activities.

SPC outputs can be used to catalyse other projects. Rather than join together many raw data sources yourself and deal with missing and messy data, you can leverage SPC's effort and well-documented schema.

A formal paper to describe SPC has been published in Environment and Planning B: Urban Analytics and City Science.

You can download this site as a PDF and find all code on Github.

This work was supported by Wave 1 of The UKRI Strategic Priorities Fund under the EPSRC Grant EP/W006022/1, particularly the “Ecosystem of Digital Twins” and “Shocks and Resilience” themes within that grant & The Alan Turing Institute

Part I

Using SPC

1 Getting started

We suggest to start by exploring one of the pre-compiled areas we have made readily available:

1. Download sample data for an area in Great Britain
2. Unpack it and open it with the web explorer

Possible next steps:

3. Learn more on how to use the data
4. If you need a custom area, build and then run SPC

1.1 What SPC does

SPC generates spatially enriched synthetic population outputs for any area that is comprised of one or more Middle-Layer Output Areas (MSOAs) in England and Wales and/or one or more Intermediary Zones (IZ) in Scotland, including Local Authority Districts - LADs. The output file generated by SPC has a granularity of Output Area (150 ±100 households). This file is structured to help other researchers or urban analysts to feed dynamic models, such as ABMs, for multiple purposes where an enriched synthetic population file is required. SPC includes a comprehensive set of variables that include sociodemographic characteristics, daily activities, and other extra data to help you model the complexity of British society.

1.2 What SPC outputs

You can see all of the per-person, household, and OA information SPC provides in the schema and data sources. We use protocol buffers to efficiently encode the data and describe its shape.

2 SPC Outputs

We provide outputs in protocol buffer format for all lieutenancy areas (more commonly known as ceremonial counties) of England, all ITL regions of Wales (international divisions based on the former unified European territorial division system) and all police divisions of Scotland for five reference years. These regions represent coherent territorial units whose scale is particularly well suited to the modelling methods used by SPC. In addition, we have included two interesting areas: the Liverpool-Manchester-Leeds arc and the Cambridge to Oxford arc. See config/ for the full list of MSOAs covered by each area.

The SPC Explorer can be used to visualise the data, understand what attributes can be obtained form SPC and get inspired about potential applications that could stem from using these outputs.

The SPC toolkit can be used to work with SPC outputs in Python using pandas or polars.

If you want to run SPC for a different list of MSOAs, see here.

The data for 2012, 2020, 2022, 2032 & 2039 are available here:

- England
- Wales
- Scotland
- North West Transpennine (2012, 2020, 2022, 2032, 2039)
- Oxford-Cambridge arc (2012, 2020, 2022, 2032, 2039)

2.1 Citing

If you use SPC code or data in your work, please cite using the Zenodo DOI (using the bottom-right tool to generate the citation).

2.2 Versioning

Over time, we may add more data to SPC or change the schema. Protocol buffers are designed to let combinations of new/old code and data files work together, but we don't intend to use this feature. We may make breaking changes, like deleting fields. We'll release a new version

of the schema and output data every time and document it here. You should depend on a specific version of the data output in your code, so new releases don't affect you until you decide to update.

- v1: released 25/04/2022, schema
- v1.1, released 27/05/2022, schema
  - added `pwkstat`, `salary_hourly`, `salary_yearly`, and `idp`
  - reorganized `Identifiers` and `Employment` attributes
  - non-breaking change added 02/08/2022: added `bmi_new` field
- v1.2, released 29/12/2022, schema
  - switched to proto2 and made some fields optional
  - adjusted some numeric enum values to match ONS
- v2, released 09/03/2023, schema
  - new per-person and per-household fields
  - various changes to existing fields (adjusting enum number, removing the BMI enum, etc)
  - adding time-use diaries
  - expanding to Wales
  - adding multiple years of output
- v2.1, released 25/07/2023, schema
  - expanding to Scotland
  - adding special area: Oxford-Cambridge arc
  - adding previously missing LADs to their counties:
    - Greater London (E09000001)
    - Cornwall (E06000053)
    - Dorset (E06000058 & E06000059)
    - Buckinghamshire (E06000060)
    - Leicestershire (E07000135)
    - Suffolk (E07000244 & E07000245)
    - Somerset (E07000246)

3 Outputs for England (Counties)

The counties of England are in this context the lieutenancy areas, often referred to as ceremonial counties. There are officially 48 of them, although we have chosen to include the City of London within Greater London in our release. Check the year you would like to explore and pick the corresponding file based on the region you are interested. Remember if you want to explore the data you can load the output in our SPC explorer.

- 2012:
  - bedfordshire.pb.gz
  - berkshire.pb.gz
  - bristol.pb.gz
  - buckinghamshire.pb.gz
  - cambridgeshire.pb.gz
  - cheshire.pb.gz
  - cornwall.pb.gz
  - cumbria.pb.gz
  - derbyshire.pb.gz
  - devon.pb.gz
  - dorset.pb.gz
  - durham.pb.gz
  - east-sussex.pb.gz
  - east-yorkshire-with-hull.pb.gz
  - essex.pb.gz
  - gloucestershire.pb.gz
  - greater-london.pb.gz (London)
  - greater-manchester.pb.gz (Manchester)
  - hampshire.pb.gz (Southampton)
  - herefordshire.pb.gz
  - hertfordshire.pb.gz
  - isle-of-wight.pb.gz
  - kent.pb.gz
  - lancashire.pb.gz
  - leicestershire.pb.gz
  - lincolnshire.pb.gz
  - merseyside.pb.gz (Liverpool)

  - norfolk.pb.gz
  - northamptonshire.pb.gz
  - northumberland.pb.gz (Newcastle)
  - north-yorkshire.pb.gz
  - nottinghamshire.pb.gz (Nottingham)
  - oxfordshire.pb.gz
  - rutland.pb.gz
  - shropshire.pb.gz
  - somerset.pb.gz
  - south-yorkshire.pb.gz (Sheffield)
  - staffordshire.pb.gz
  - suffolk.pb.gz
  - surrey.pb.gz
  - tyne-and-wear.pb.gz
  - warwickshire.pb.gz
  - west-midlands.pb.gz (Birmingham)
  - west-sussex.pb.gz
  - west-yorkshire.pb.gz (Leeds)
  - wiltshire.pb.gz
  - worcestershire.pb.gz

- 2020:

  - bedfordshire.pb.gz
  - berkshire.pb.gz
  - bristol.pb.gz
  - buckinghamshire.pb.gz
  - cambridgeshire.pb.gz
  - cheshire.pb.gz
  - cornwall.pb.gz
  - cumbria.pb.gz
  - derbyshire.pb.gz
  - dorset.pb.gz
  - devon.pb.gz
  - durham.pb.gz
  - east-sussex.pb.gz
  - east-yorkshire-with-hull.pb.gz
  - essex.pb.gz
  - gloucestershire.pb.gz
  - greater-london.pb.gz (London)
  - greater-manchester.pb.gz (Manchester)
  - hampshire.pb.gz (Southampton)
  - herefordshire.pb.gz

  - hertfordshire.pb.gz
  - isle-of-wight.pb.gz
  - kent.pb.gz
  - lancashire.pb.gz
  - leicestershire.pb.gz
  - lincolnshire.pb.gz
  - merseyside.pb.gz (Liverpool)
  - norfolk.pb.gz
  - northamptonshire.pb.gz
  - northumberland.pb.gz (Newcastle)
  - north-yorkshire.pb.gz
  - nottinghamshire.pb.gz (Nottingham)
  - oxfordshire.pb.gz
  - rutland.pb.gz
  - shropshire.pb.gz
  - somerset.pb.gz
  - south-yorkshire.pb.gz (Sheffield)
  - staffordshire.pb.gz
  - suffolk.pb.gz
  - surrey.pb.gz
  - tyne-and-wear.pb.gz
  - warwickshire.pb.gz
  - west-midlands.pb.gz (Birmingham)
  - west-sussex.pb.gz
  - west-yorkshire.pb.gz (Leeds)
  - wiltshire.pb.gz
  - worcestershire.pb.gz

- 2022:

  - bedfordshire.pb.gz
  - berkshire.pb.gz
  - bristol.pb.gz
  - buckinghamshire.pb.gz
  - cambridgeshire.pb.gz
  - cheshire.pb.gz
  - cornwall.pb.gz
  - cumbria.pb.gz
  - derbyshire.pb.gz
  - dorset.pb.gz
  - devon.pb.gz
  - durham.pb.gz
  - east-sussex.pb.gz

  - east-yorkshire-with-hull.pb.gz
  - essex.pb.gz
  - gloucestershire.pb.gz
  - greater-london.pb.gz (London)
  - greater-manchester.pb.gz (Manchester)
  - hampshire.pb.gz (Southampton)
  - herefordshire.pb.gz
  - hertfordshire.pb.gz
  - isle-of-wight.pb.gz
  - kent.pb.gz
  - lancashire.pb.gz
  - leicestershire.pb.gz
  - lincolnshire.pb.gz
  - merseyside.pb.gz (Liverpool)
  - norfolk.pb.gz
  - northamptonshire.pb.gz
  - northumberland.pb.gz (Newcastle)
  - north-yorkshire.pb.gz
  - nottinghamshire.pb.gz (Nottingham)
  - oxfordshire.pb.gz
  - rutland.pb.gz
  - shropshire.pb.gz
  - somerset.pb.gz
  - south-yorkshire.pb.gz (Sheffield)
  - staffordshire.pb.gz
  - suffolk.pb.gz
  - surrey.pb.gz
  - tyne-and-wear.pb.gz
  - warwickshire.pb.gz
  - west-midlands.pb.gz (Birmingham)
  - west-sussex.pb.gz
  - west-yorkshire.pb.gz (Leeds)
  - wiltshire.pb.gz
  - worcestershire.pb.gz

- 2032:

  - bedfordshire.pb.gz
  - berkshire.pb.gz
  - bristol.pb.gz
  - buckinghamshire.pb.gz
  - cambridgeshire.pb.gz
  - cheshire.pb.gz

  - cornwall.pb.gz
  - cumbria.pb.gz
  - derbyshire.pb.gz
  - devon.pb.gz
  - dorset.pb.gz
  - durham.pb.gz
  - east-sussex.pb.gz
  - east-yorkshire-with-hull.pb.gz
  - essex.pb.gz
  - gloucestershire.pb.gz
  - greater-london.pb.gz (London)
  - greater-manchester.pb.gz (Manchester)
  - hampshire.pb.gz (Southampton)
  - herefordshire.pb.gz
  - hertfordshire.pb.gz
  - isle-of-wight.pb.gz
  - kent.pb.gz
  - lancashire.pb.gz
  - leicestershire.pb.gz
  - lincolnshire.pb.gz
  - merseyside.pb.gz (Liverpool)
  - norfolk.pb.gz
  - northamptonshire.pb.gz
  - northumberland.pb.gz (Newcastle)
  - north-yorkshire.pb.gz
  - nottinghamshire.pb.gz (Nottingham)
  - oxfordshire.pb.gz
  - rutland.pb.gz
  - shropshire.pb.gz
  - somerset.pb.gz
  - south-yorkshire.pb.gz (Sheffield)
  - staffordshire.pb.gz
  - suffolk.pb.gz
  - surrey.pb.gz
  - tyne-and-wear.pb.gz
  - warwickshire.pb.gz
  - west-midlands.pb.gz (Birmingham)
  - west-sussex.pb.gz
  - west-yorkshire.pb.gz (Leeds)
  - wiltshire.pb.gz
  - worcestershire.pb.gz

- 2039:

- bedfordshire.pb.gz
- berkshire.pb.gz
- bristol.pb.gz
- buckinghamshire.pb.gz
- cambridgeshire.pb.gz
- cheshire.pb.gz
- cornwall.pb.gz
- cumbria.pb.gz
- derbyshire.pb.gz
- devon.pb.gz
- dorset.pb.gz
- durham.pb.gz
- east-sussex.pb.gz
- east-yorkshire-with-hull.pb.gz
- essex.pb.gz
- gloucestershire.pb.gz
- greater-london.pb.gz (London)
- greater-manchester.pb.gz (Manchester)
- hampshire.pb.gz (Southampton)
- herefordshire.pb.gz
- hertfordshire.pb.gz
- isle-of-wight.pb.gz
- kent.pb.gz
- lancashire.pb.gz
- leicestershire.pb.gz
- lincolnshire.pb.gz
- merseyside.pb.gz (Liverpool)
- norfolk.pb.gz
- northamptonshire.pb.gz
- northumberland.pb.gz (Newcastle)
- north-yorkshire.pb.gz
- nottinghamshire.pb.gz (Nottingham)
- oxfordshire.pb.gz
- rutland.pb.gz
- shropshire.pb.gz
- somerset.pb.gz
- south-yorkshire.pb.gz (Sheffield)
- staffordshire.pb.gz
- suffolk.pb.gz
- surrey.pb.gz
- tyne-and-wear.pb.gz
- warwickshire.pb.gz
- west-midlands.pb.gz (Birmingham)

- west-sussex.pb.gz
- west-yorkshire.pb.gz (Leeds)
- wiltshire.pb.gz
- worcestershire.pb.gz

3.1 Citing

If you use SPC code or data in your work, please cite using the Zenodo DOI (using the bottom-right tool to generate the citation).

4 Outputs for Wales (ITL regions)

International Territorial Level (ITL) regions are a post-brexit renaming of the former Nomenclature of Territorial Units for Statistics (NUTS) regions. In wales, the level 3 represents a grouping of the 22 unitary districts into 12 regions. Check the year you would like to explore and pick the corresponding file based on the region you are interested. Remember if you want to explore the data you can load the output in our SPC explorer.

- 2012:
  - bridgend-and-neath-port-talbot.pb.gz
  - cardiff-and-vale-of-glamorgan.pb.gz
  - central-valleys.pb.gz
  - conwy-and-denbighshire.pb.gz
  - flintshire-and-wrexham.pb.gz
  - gwent-valleys.pb.gz
  - gwynedd.pb.gz
  - isle-of-anglesey.pb.gz
  - monmouthshire-and-newport.pb.gz
  - powys.pb.gz
  - south-west-wales.pb.gz
  - swansea.pb.gz
- 2020:
  - bridgend-and-neath-port-talbot.pb.gz
  - cardiff-and-vale-of-glamorgan.pb.gz
  - central-valleys.pb.gz
  - conwy-and-denbighshire.pb.gz
  - flintshire-and-wrexham.pb.gz
  - gwent-valleys.pb.gz
  - gwynedd.pb.gz
  - isle-of-anglesey.pb.gz
  - monmouthshire-and-newport.pb.gz
  - powys.pb.gz
  - south-west-wales.pb.gz
  - swansea.pb.gz
- 2022:

  - bridgend-and-neath-port-talbot.pb.gz
  - cardiff-and-vale-of-glamorgan.pb.gz
  - central-valleys.pb.gz
  - conwy-and-denbighshire.pb.gz
  - flintshire-and-wrexham.pb.gz
  - gwent-valleys.pb.gz
  - gwynedd.pb.gz
  - isle-of-anglesey.pb.gz
  - monmouthshire-and-newport.pb.gz
  - powys.pb.gz
  - south-west-wales.pb.gz
  - swansea.pb.gz
- 2032:
  - bridgend-and-neath-port-talbot.pb.gz
  - cardiff-and-vale-of-glamorgan.pb.gz
  - central-valleys.pb.gz
  - conwy-and-denbighshire.pb.gz
  - flintshire-and-wrexham.pb.gz
  - gwent-valleys.pb.gz
  - gwynedd.pb.gz
  - isle-of-anglesey.pb.gz
  - monmouthshire-and-newport.pb.gz
  - powys.pb.gz
  - south-west-wales.pb.gz
  - swansea.pb.gz
- 2039:
  - bridgend-and-neath-port-talbot.pb.gz
  - cardiff-and-vale-of-glamorgan.pb.gz
  - central-valleys.pb.gz
  - conwy-and-denbighshire.pb.gz
  - flintshire-and-wrexham.pb.gz
  - gwent-valleys.pb.gz
  - gwynedd.pb.gz
  - isle-of-anglesey.pb.gz
  - monmouthshire-and-newport.pb.gz
  - powys.pb.gz
  - south-west-wales.pb.gz
  - swansea.pb.gz

4.1 Citing

If you use SPC code or data in your work, please cite using the Zenodo DOI (using the bottom-right tool to generate the citation).

5 Outputs for Scotland (Police Divisions)

Police divisions are a convenient grouping of unitary districts. Check the year you would like to explore and pick the corresponding file based on the region you are interested. Remember if you want to explore the data you can load the output in our SPC explorer.

- 2012:
  - argyll-and-west-dunbartonshire.pb.gz
  - ayrshire.pb.gz
  - dumfries-and-galloway.pb.gz
  - edinburgh.pb.gz
  - fife.pb.gz
  - forth-valley.pb.gz
  - greater-glasgow.pb.gz
  - highlands-and-islands.pb.gz
  - lanarkshire.pb.gz
  - north-east.pb.gz
  - renfrewshire-and-inverclyde.pb.gz
  - tayside.pb.gz
  - the-lothians-and-scottish-borders.pb.gz
- 2020:
  - argyll-and-west-dunbartonshire.pb.gz
  - ayrshire.pb.gz
  - dumfries-and-galloway.pb.gz
  - edinburgh.pb.gz
  - fife.pb.gz
  - forth-valley.pb.gz
  - greater-glasgow.pb.gz
  - highlands-and-islands.pb.gz
  - lanarkshire.pb.gz
  - north-east.pb.gz
  - renfrewshire-and-inverclyde.pb.gz
  - tayside.pb.gz
  - the-lothians-and-scottish-borders.pb.gz
- 2022:

  - argyll-and-west-dunbartonshire.pb.gz
  - ayrshire.pb.gz
  - dumfries-and-galloway.pb.gz
  - edinburgh.pb.gz
  - fife.pb.gz
  - forth-valley.pb.gz
  - greater-glasgow.pb.gz
  - highlands-and-islands.pb.gz
  - lanarkshire.pb.gz
  - north-east.pb.gz
  - renfrewshire-and-inverclyde.pb.gz
  - tayside.pb.gz
  - the-lothians-and-scottish-borders.pb.gz
- 2032:
  - argyll-and-west-dunbartonshire.pb.gz
  - ayrshire.pb.gz
  - dumfries-and-galloway.pb.gz
  - edinburgh.pb.gz
  - fife.pb.gz
  - forth-valley.pb.gz
  - greater-glasgow.pb.gz
  - highlands-and-islands.pb.gz
  - lanarkshire.pb.gz
  - north-east.pb.gz
  - renfrewshire-and-inverclyde.pb.gz
  - tayside.pb.gz
  - the-lothians-and-scottish-borders.pb.gz
- 2039:
  - argyll-and-west-dunbartonshire.pb.gz
  - ayrshire.pb.gz
  - dumfries-and-galloway.pb.gz
  - edinburgh.pb.gz
  - fife.pb.gz
  - forth-valley.pb.gz
  - greater-glasgow.pb.gz
  - highlands-and-islands.pb.gz
  - lanarkshire.pb.gz
  - north-east.pb.gz
  - renfrewshire-and-inverclyde.pb.gz
  - tayside.pb.gz
  - the-lothians-and-scottish-borders.pb.gz

5.1 Citing

If you use SPC code or data in your work, please cite using the Zenodo DOI (using the bottom-right tool to generate the citation).

6 How to use the output file

Once you download or generate an SPC output file for your study area, how do you use it? Each study area consists of one `.pb` or protocol buffer file. This file efficiently encodes data following this schema. Read more about what data is contained in the output.

You can read the "protobuf" (shorthand for a protocol buffer file) in any supported language, and then extract and transform just the parts of the data you want for your model.

We have examples for Python below, but feel free to request other languages.

6.1 Javascript

We have a web app using Svelte to interactively explore SPC data. Its source code is great reference for how to use the proto output.

6.2 Python

To work with SPC outputs in Python, we recommend using the SPC toolkit.

6.2.1 Install

The package can be installed with `pip` from git with:

```
pip install 'git+https://github.com/alan-turing-institute/uatk-spc.git#subdirectory=python'
```

or with Poetry:

```
poetry add 'git+https://github.com/alan-turing-institute/uatk-spc.git#subdirectory=python'
```

To include extra dependencies `dev` and `examples` for running tests and example notebooks:

```
pip install 'uatk-spc[dev,examples] @ git+https://github.com/alan-turing-institute/uatk-spc.g
```

or with Poetry:

```
poetry add 'git+https://github.com/alan-turing-institute/uatk-spc.git#subdirectory=python'
```

6.2.2 Reading and Building

The two classes provided by the toolkit are a `Reader` (reads all population fields: `people`, `households`, `venues_per_activity`, `time_use_diaries` and `info_per_msoa`) and a `Builder` (extends the `Reader` to include methods for combining the fields on top a people dataframe to build a single final dataframe).

An example of using the `Reader` is shown below:

```
# Import package
from uatk_spc import Reader
# Pick a region with SPC output saved
(region, path) = "rutland", "data/output/England/2020/"
# Read from parquet and JSON
population = Reader(path, region, backend="polars")
# Or directly from a filepath to a gzip archive
population = Reader(
    filepath="https://ramp0storage.blob.core.windows.net/test-spc-output/test_region.tar.gz"
    backend="polars"
)
# Write people to csv
population.people.to_pandas().to_csv("people.csv", index=False)
```

And an example of using the `Builder`:

```
# Import Builder
from uatk_spc import Builder
(region, path) = "rutland", "../data/output/England/2020/"
# Build population from people and households and unnest "health" and "details"
population = (
  Builder(path, region, backend="polars")
  .add_households()
  .unnest(["health", "details"])
  .build()
)
```

```
# Write combined people and household dataframe to csv
population.to_pandas().to_csv("people_and_households.csv", index=False)
```

For further examples of using the `Reader` and `Builder` to analyse SPC outputs, see the examples folder.

6.2.3 Converting .pb file to JSON format

To interactively explore the data, viewing JSON is much easier. It shows the same structure as the protobuf, but in a human-readable text format. The example below uses a small Python script:

```
# Download a file
wget https://ramp0storage.blob.core.windows.net/spc-output/v2.1/England/2020/rutland.pb.gz
# Uncompress
gunzip rutland.pb.gz
# Convert the .pb to JSON
# - without poetry
python python/uatk_spc/scripts.py --input-path data/output/England/2020/rutland.pb > rutland
# - with poetry
cd python && poetry run spc_to_json --input-path ../data/output/England/2020/rutland.pb > ru
# View the output
less rutland.json
```

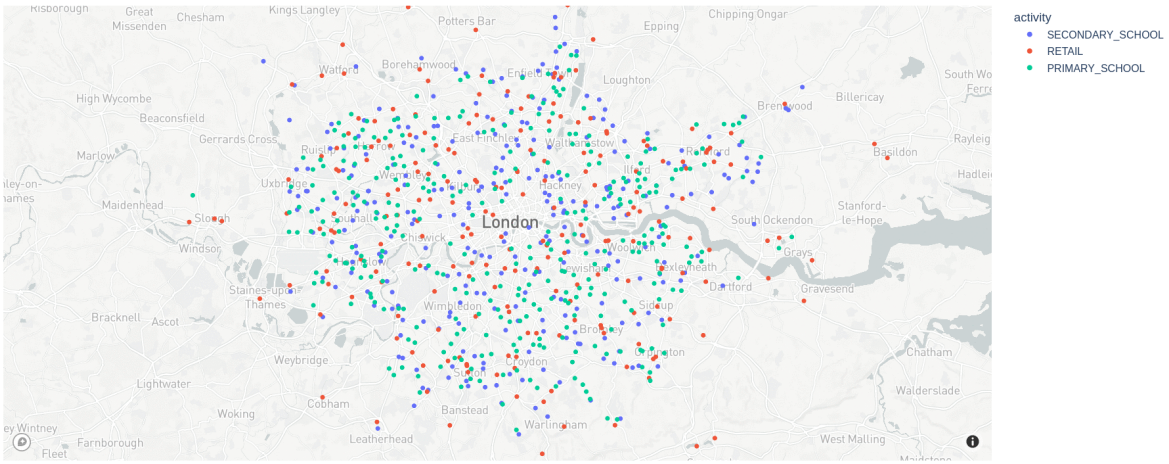
6.2.4 Converting to numpy arrays

The ASPICS project simulates the spread of COVID through a population. The code uses numpy, and this script converts the protobuf to a bunch of different numpy arrays.

Note the ASPICS code doesn't keep using the generated Python protobuf classes for the rest of the pipeline. Data frames and numpy arrays may be more familiar and appropriate. The protobuf is a format optimized for reading and writing; you don't need to use it throughout all of your model code.

6.2.5 Visualizing venues

Use this script to read a protobuf file, then draws a dot for every venue, color-coded by activity.


activity
SECONDARY_SCHOOL
RETAIL
PRIMARY_SCHOOL
London
Potters Bar
Epping
Chipping Ongar
Watford
Borehamwood
Enfield
Loughton
High Wycombe
Beaconsfield
Billericay
Brentwood
Gerrards Cross
Marlow
Maidenhead
Uxbridge
Hackney
Ilford
Romford
Basildon
Stanford-le-Hope
South Ockendon
Slough
Windsor
Chiswick
Woolwich
Grays
Dartford
Gravesend
Staines-upon-Thames
Wimbledon
Bracknell
Ascot
Bromley
Sidcup
Chatham
Weybridge
Croydon
Lightwater
Walderslade
Cobham
Woking
Banstead
Warlingham
Leatherhead
Farnborough
Fleet
West Malling

7 Full tool installation

This guide allows you to install the full SPC tool to run a custom area.

7.1 Dependencies

- **Rust**: The latest stable version of Rust: https://www.rust-lang.org/tools/install

7.2 Compiling SPC

```
git clone https://github.com/alan-turing-institute/uatk-spc/
cd uatk-spc
# The next command will take a few minutes the first time you do it, to build external depend
cargo build --release
```

7.3 Troubleshooting downloading

If you get an error `No such file or directory (os error 2)` it might be because a previous attempt to run SPC failed, and some necessary files were not fully downloaded. In these cases you could try deleting the `data/raw_data` directory and then running SPC again. It should automatically try to download the big files again.

If you have trouble downloading any of the large files, you can download them manually. The logs will contain a line such as `Downloading https://ramp0storage.blob.core.windows.net/nationaldata/ to data/raw_data/nationaldata/QUANT_RAMP_spc.tar.gz`. This tells you the URL to retrieve, and where to put the output file. Note that SPC won't attempt to download files if they already exist, so if you wind up with a partially downloaded file, you have to manually remove it.

8 Running a custom area

If the area you want to model isn't already generated, then you can follow this guide to run SPC on a custom area. You must first compile SPC.

8.1 Specifying the area

SPC takes a newline-separated list of MSOAs in the `config/` directory as input, like this. You can generate this list from a LAD (local authority district). From the main SPC directory, run `python scripts/select_msoas.py`. Refer to `data/raw_data/referencedata/lookUp.csv` (only available after running SPC once) for all geographies available.

This script will create a new file, `config/your_region.txt`.

8.2 Run SPC for the new area

From the main directory, just run:

```
cargo run --release -- config/your_region.txt
```

This will download some large files the first time. You'll wind up with `data/output/your_region.pb` as output, as well as lots of intermediate files in `data/raw_data/`. The next time you run this command (even on a different study area), it should go much faster.

8.3 (Optional) Output to parquet format

As an alternative to protobuf outputs, you can also run the SPC and choose to output to parquet format instead by running:

```
cargo run --release -- config/your_region.txt --output-formats parquet
```

or if both protobuf and parquet are required, change the flag to `--output-formats protobuf,parquet`.

8.4 (Optional) run SPC for lots of areas

If you want to run the program over lots of areas at once and are using Mac/Linux, you can use a `for` loop in a terminal to repeatedly run SPC over all files in the config directory. For example, this will run SPC on all `.txt` files in the `config` directory:

```
for file in config/*.csv; do cargo run --release -- config/$file; done
```

8.5 Using the output

After you generate the files, see here for how to use them in your project.

If you use SPC code or data in your work, please cite using the Zenodo DOI (using the bottom-right tool to generate the citation).

Part II

Understanding SPC

9 Introduction

SPC is divided into two phases. The data preparation phase relies on scripts that only need to be run once. It outputs a postprocessed version of all the raw data sources that allows the model to run smoothly on custom areas. The second phase involves the user choosing a custom area and launching a simulation. It pulls the relevant datasets among the data prepared by the first phase, calculates the different daily activities and formats the results into a single protocol buffer file.

In this section, you can find:

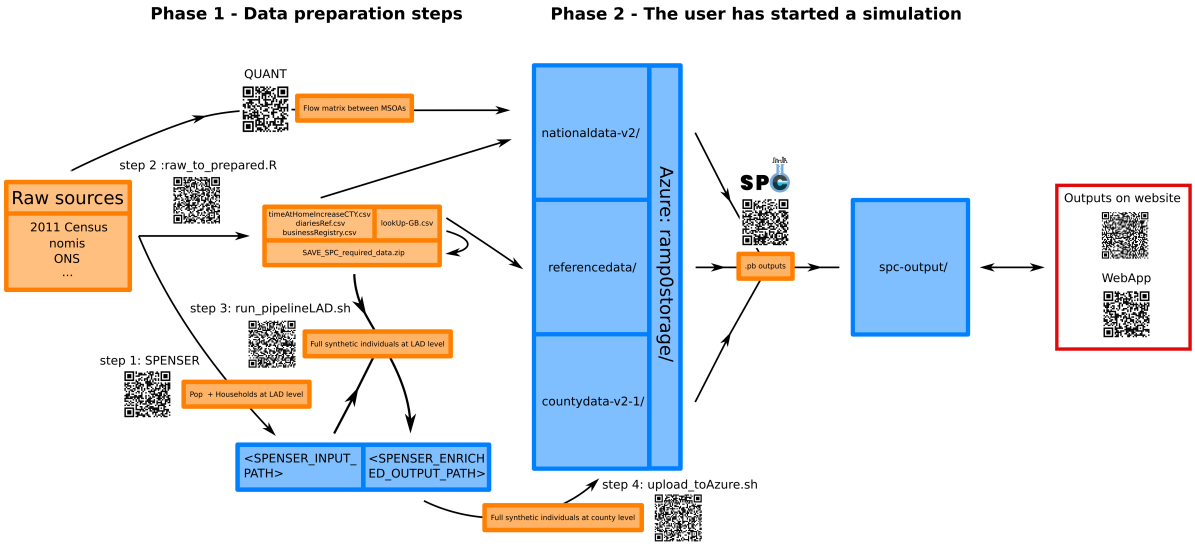
- A step by step description of each element of the SPC pipeline
- The concepts supporting the modelling methods
- A description of the schema of the protocol buffer
- A list of the data sources used to create each data field

Note that due to the large impact each section has on the choices made in other sections, it may be necessary to frequently segue between sections to get a precise understanding of the model.

10 The SPC pipeline

We provide in this document a step by step approach to running the entire SPC pipeline. This pipeline comprises the following steps.

- **Phase 1 - Data preparation**
  1. The SPENSER model creates a synthetic population with basic demographic information for all of GB.
  2. A script downloads and prepares data from various public sources that will be used throughout the model.
  3. The outputs of SPENSER are enriched using some of the outputs from step 2.
  4. The resulting outputs are uploaded as `.csv` files to a dedicated Azure repository.
- **Phase 2 - The user has selected a study area and started a simulation**
  1. All the data relevant to the study area are pulled from Azure.
  2. Individuals are assigned a single education destination (if under 16 y.o.) and several potential retail destinations, according to a local version of the QUANT model.
  3. Individuals are assigned a single workplace destination (if above 16 y.o), according to the method described here.
  4. The population, its activities and an optional lockdown modelling are gathered into a single `.pb` file that can be visualised with the SPC explorer.



We now explain how to run each step. The theoretical concepts supporting the modelling are presented here. The different data sources are detailed here.

10.1 Phase 1: Data preparation

10.1.1 SPENSER

The original SPENSER (Synthetic Population Estimation and Scenario Projection) model is made up of 5 different GitHub repositories, operating specific parts of the simulation of a synthetic population (gathering the data from ONS, creating individuals, creating households, assigning individuals to households and projecting the population to future years). We use this modified version with instructions for running the full pipeline on a single machine.

The SPENSER microsimulation is split into three steps:

1. Household synthesis: households are synthesised for a base year (2011) from census data at OA resolution. These households are then sequentially synthesised for subsequent years using household forecasts.
2. Population synthesis: people are sequentially synthesised using marginal census data on gender, age and ethnicity at MSOA resolution for 2011, with population projections used to derive the marginals beyond the reference census year.
3. Assignment: for a given year, a synthesised household (from step 1) is assigned to each member of the synthesised population (from step 2), while a "household representative person" from the synthesised population (from step 2) is assigned to each synthesised household (from step 1).

The result of SPENSER is two separate datasets and a merging key: one dataset for individuals, accurate at MSOA level and containing the `sex`, `age` and `ethnicity` fields; and one for households, accurate at OA level and containing the `OA11CD`, `HOUSE_nssec8`, `House_type`, `HOUSE_typeCommunal`, `HOUSE_NRooms`, `HOUSE_centralHeat`, `HOUSE_tenure` and `HOUSE_NCars` fields.

10.1.2 Downloading and preparation of public data from various sources

Instructions to run this step from the source code can be found under Step 1: Curate public data from diverse sources. More information on the various data sources can be found here. The result is a set of data files, some of which will be merged with the outputs from SPENSER during the next step, containing:

- NSSEC8 distributions among the population of England and Wales by age group and sex at MSOA level (`NSSEC8_EW_F_16to24_CLEAN.csv`, etc.) and among the total population of Scotland by age group, sex and ethnicity (`NSSECS_CLEAN.csv`)

- A combined extract from the three latest GB Health Surveys (`HSComplete.csv`)
- An extract from the UK Time Use Survey 2015 (`indivTUS.csv`)
- A file containing a set of coefficients to estimate the average BMI of individuals in England depending on their age, sex and ethnicity (`BMIdMean.csv`) and a file containing coefficients to obtain the equivalent average BMI in Scotland and Wales (`BMIdiff.csv`)
- Coefficients to estimate the hourly salary of an employee in England depending on their home region, sex, part-time/full-time status, age and SOC category (`coefFFT.csv`, etc. and `ageRescaleFFT.csv`, etc.).
- Coefficients to estimate the numbers of hours worked corresponding to the criteria mentioned above (`meanHoursFFT.csv`, etc. and `distribHours.csv`)
- Centroid coordinates of Output areas in GB (`OACentroids.csv`)

In addition, four files to be used by the second phase of the model are created:

- `diariesRef.csv` contains diaries of typical days extracted from the UK Time Use Survey
- `businessRegistry.csv` contains a list of all individual workplaces in GB
- `timeAtHomeIncreaseCTY.csv` contains a reduction in time spent away from home during the pandemic according to Google Mobility reports
- `lookUp-GB.csv` is a comprehensive lookup table between GB geographies, including name variants used by Google and OSM and local file names for storage within Azure

To understand the methods supporting the creation of these files, we refer the reader to the modelling methods section.

10.1.3 Enriching SPENSER

Instructions to run this step can be found under Step 2: Add to SPENSER. Line numbers quoted in the following refer to this script.

Once merged into one dataset according to the matching key (l. 13-49), the SPENSER data is enriched with the outputs of the previous step. An individual among those sharing the same 5-year age group (extra details for under 18) and sex is drawn (uniform distribution) from the participants of the Health Survey (l. 56-72). This adds the `id_HS`, `HEALTH_diabetes`, `HEALTH_bloodpressure`, `HEALTH_cvd`, `HEALTH_NMedicines`, `HEALTH_selfAssessed` and `HEALTH_lifeSat` fields. This join is not spatially differentiated and other potential matching criteria (such as ethnicity and nssec8) were not retained due to a lack of representativity for some groups inside the survey. The BMI field is then added l. 74-89, according to this method.

Each individual that is not a head of household is assigned an nssec8 category (l. 96-108). The probability distribution is given by nssec8 category distributions among the general population by sex and age groups according to ONS data (DC6114EW and DC6206SC datasets). An individual among those sharing the same 5-year age group, sex and nssec8 category is drawn (uniform distribution) from the participants of the UK Time Use Survey (l. 111-125). This

adds the `id_TUS_hh`, `id_TUS_p`, `pwkstat`, `soc2010`, `sic1d2007`, `sic2d2007`, `netPayWeekly` and `workedHoursWeekly` fields. Note that the `netPayWeekly` and `workedHoursWeekly` fields had a low response rate among participants of the survey. For that reason, we have added a much more detailed modelling of income, that includes spatial differences at region level (l. 130-140).

Coordinates of the centroids of the OAs where the household's home is located are finally added l. 152-156.

10.1.4 Azure upload

Following enrichment, a final step involves grouping LADs into counties and uploading to an Azure container for use as input for Phase 2 below.

10.2 Phase 2: Running SPC for a specific study area

This part is corresponding to the scripts written in Rust. Instructions can be found here.

11 Modelling methods

We present here the theoretical principles behind the modelling done in SPC and point to the parts of the code where they are implemented.

11.1 SPENSER and QUANT

The generation of the population data by SPENSER (Synthetic Population Estimation and Scenario Projection) model and the modelling of trips to schools and retail by QUANT are detailed in

> Lomax N et al. An Open-Source Model for Projecting Small Area Demographic and Land-Use Change. Geographical analysis, 54(3), 599-622 (2022). (DOI)

and

> Spooner F et al. A dynamic microsimulation model for epidemics. Soc Sci Med., 291:114461 (2021). (DOI)

11.2 BMI estimation

Body Max Index (BMI) is calculated for each individual from the Health Survey for England 2019 (access needs to be requested to the UK Data Service). This calculation is independent from the matching with the Heath Survey that happens during the data preparation step, therefore the BMI values will not match the ones that could be obtained from the Health Survey identifiers included in the output. As the BMI variable is not necessarily independent from the other health variables (diabetes etc.), the new variable should only be used for studies where all other variables are considered equal. The new variable is continuous (a float) instead of categorical.

According to the HSE 2019, the distribution of BMI values should follow figure 1. The socio-economic category variable was discarded for the modelling as it is not independent from the other variables, and "mixed" and "other" ethnicity categories have been merged due to small sample sizes.

Figure 1. BMI per age. Columns represent ethnicity (White, Black, Asian, Other), and the rows sex (female, male).

The distribution for each age group is a gamma distribution. See figure 2.

Figure 2. Distribution of BMI values for white females aged 30-34.

Due to small sample sizes, the BMI is calculated for each individual depending on their age according to a gamma distribution whose mean is the mean for the corresponding age, sex and ethnicity (thick line in figure 1), but whose variance is only determined by the total variance by sex and ethnicity. The resulting BMI values were validated for Bedfordshire, and correlations of 0.93 and 0.97 were found between the mean and variance of the modelled data compared to those for the reference HSE 2019 data. See figure 3. The distribution per age, as in figure 1, were also validated.

Figure 3. Modelled mean and variance compared to the reference mean and variance from the HSE 2019 data for each of the eight categories of figure 1.

The R code for this modelling are l. 239-406 of this script, and the validation is included in the legacy version, here.

11.3 Income data

This modelling is based on the 2020 revised edition of the Earnings and hours worked, region by occupation by four-digit SOC: ASHE Table 15 database from ONS. Some percentiles for employees' gross hourly salaries are provided for each full-time and part-time job according to their four-digit SOC classification per region, and separated by sex. It is supported by this script.

11.3.1 Methods

The data are far from complete (only about 15% of all possible values), especially for the highest deciles. We found that the missing values amongst the partially filled SOCs could be estimated by interpolating an order 3 polynomial to the existing values. We found that the order 3 polynomials were a good fit for most categories (93.11%). SOCs with too many missing values are given the value for the category that is immediately higher in the SOC hierarchy. For some jobs, the highest percentiles seem capped, making the polynomial fitting fail. In that case, we have replaced the unknown values with the highest known value (as there is no clear and systematic fitting for these special cases). In addition, the highest decile is never detailed in the data, which means that the highest salaries are underestimated in the model (and exceptionally high salaries are not present). The result of this phase is four tables {male full-time, male part-time, female full-time, female part-time} containing the coefficients of the

fitted order 3 polynomials, with an optional cap when relevant. This step is done in section 1 of the script.

A percentile is chosen randomly (uniformly) for each individual in England, and the salary is then deduced according to their full-time/part-time status, region, sex and SOC category. Age data from ONS are then integrated. Part of the differences that can be observed between different age groups is already taken into account through the SOC variable, since it is expected to evolve throughout an individual's career. To avoid counting this dependency twice, we compute the residual between the results of the initial modelling that does not take into account age and the expected results by age group according to the data. We then deduce a function that modifies a posteriori the estimated salary of an individual depending on their age, so that the salaries sum correctly by age groups. This step is done in sections 2 to 4 of the script.

To get the number of hours worked per week, we also use the ASHE Table 15. Since only minimal differences are observed between SOC categories, we simply complete missing values by approximating them by values of the category that is immediately higher within the SOC hierarchy. This step is done in section 5 of the script.

When added to the SPC population data, a basic hourly salary column is added, as well as a corresponding annual salary deduced from the number of worked hours. In addition, we repeat this process for all individuals that are categorised as 'Self-employed' or 'Employee unspecified' by the Time Use Survey matching„ as if they were full time employees. These values are recorded in the columns `IncomeHAsIF` and `IncomeYAsIf`.

11.3.2 Comparison to reference values from ONS

We compare the results of the modelling to the raw datasets from ONS.

- `Mod` for modelled
- `M` for male
- `F` for female
- `H` for hourly gross salary
- `Y` for annual gross salary
- `FT` for full-Time
- `PT` for part-Time
- Only individuals recorded as employees (i.e. not self-employed) are taken into account in this section.

Number of employees per sex and full-time/part-time classification

The numbers given by ONS vary from dataset to dataset and are reported by ONS as indicative only. For the modelled values, we give the total number of individuals with a non-zero salary in each category.

Variable	All	FT	PT	M	M FT	M PT	F	F FT	F PT
ONS tot	22-26k	16-19k	6-8k	11-13k	9-11k	1.5-2k	11-13k	6.5-7.5k	4.5-5.5k
Mod tot H	23.1k	18.5k	4.6k	11.8k	11k	0.8k	11.3k	7.5k	3.8k
Mod tot Y	17.6k	14.8k	2.8k	9.4k	8.9k	0.5k	8.2k	5.9k	2.3k

A significant number of individuals listed as working either full or part time have 0 effective worked hours per day according to the Time Use Survey matching. In those cases, an hourly salary is modelled depending on their SOC, region and sex, as for any other employee, but the annual salary will be displayed as 0. It is possible to estimate the likely true number of hours worked from the same ONS dataset (Table 15.9a: Paid hours worked - Total 2020), also depending on their sex, soc and region. This calculation has been added to the "As If" column.

Hourly gross salary per sex and full-time/part-time classification

Variable	All	FT	PT	M	M FT	M PT	F	F FT	F PT
ONS mean	17.63	18.32	13.93	18.81	19.12	14.69	16.19	17.08	13.68
ONS median	13.71	15.15	10.38	14.84	15.58	10.12	12.58	14.42	10.47
Mod mean	16.45	17.19	13.45	17.50	17.84	12.75	15.35	16.23	13.60
Mod median	13.55	14.46	10.23	14.27	14.72	9.16	12.79	14.12	10.51

The median values are quite close to the ONS values, but the mean values are always lower. This is expected, see the description of the modelling above.

Annual gross salary per sex and full-time/part-time classification

Only values > 0 are retained for these calculations.

Variable	All	FT	PT	M	M FT	M PT	F	F FT	F PT
ONS mean	31,646	38,552	13,819	38,421	42,072	14,796	24,871	33,253	13,512
ONS median	25,886	31,487	11,240	31,393	33,915	10,883	20,614	28,002	4,743
Mod mean	34,317	36,595	22,257	37,574	38,496	20,698	30,594	33,729	22,585
Mod median	28,713	30,942	17,928	31,404	32,382	17,382	25,875	29,028	18,137

The average salary for part-time employees is correct when values equal to 0 are taken into account. This suggests that the total number of hours worked for part-time employees is

correct, but the way they are distributed among individuals is not. It could be due to the TUS taking a snapshot of the situation during a particular week, rather than averaging their data over the year. It appears that the TUS matching also overestimates the average number of hours worked for female employees.

Regional differences (hourly gross salary)

Region	East	East Mid-lands	London	North East	North West	South East	South West	West Mid-lands	Yorkshire and The Humber
ONS mean	16.74	15.87	23.78	15.69	16.36	17.88	16.36	16.34	15.76
ONS median	13.28	12.65	18.30	12.40	12.90	14.33	12.74	12.92	12.46
Mod mean	16.67	15.29	19.39	15.05	15.22	17.34	15.92	15.47	14.41
Mod median	13.69	12.79	16.25	12.42	12.44	14.84	13.35	12.64	12.44

The pearson correlations for mean and median between the modelled and raw values are 0.92 and and 0.93.

Hourly gross salary per one-digit SOC

1d SOC	1	2	3	4	5	6	7	8	9
ONS mean	26.77	23.38	18.29	13.42	13.35	10.87	10.94	12.23	10.77
ONS median	20.96	21.34	15.66	11.54	12.04	10.08	9.52	10.93	9.22
Mod mean	21.52	22.14	16.00	12.76	12.55	10.49	10.50	12.05	9.87
Mod median	17.22	20.66	14.12	11.46	11.34	9.71	9.59	10.82	9.12

1. Managers, directors and senior officials
2. Professional occupations
3. Associate professional and technical occupations
4. Administrative and secretarial occupations
5. Skilled trades occupations
6. Caring, leisure and other service occupations
7. Sales and customer service occupations
8. Process, plant and machine operatives
9. Elementary occupations.

The Pearson correlations for mean and median between the modelled and raw values are 0.98 and 0.98.

Hourly gross salary per age

The reference for this table is: Table 6.5a Hourly pay - Gross 2020

Table before weighting by age:

Age	16-17	18-21	22-29	30-39	40-49	50-59	60+
ONS mean	7.21	9.59	14.09	18.13	20.04	19.12	16.32
ONS median	6.36	9.00	12.26	15.08	15.89	14.39	12.17
Mod mean	12.77	14.96	16.33	16.93	16.83	16.66	16.29
Mod median	10.93	12.71	13.88	14.02	13.96	13.85	13.65

The Pearson correlations for mean and median between the modelled and raw values are 0.92 and 0.92.

Table after weighting by age:

Age	16-17	18-21	22-29	30-39	40-49	50-59	60+
ONS mean	7.21	9.59	14.09	18.13	20.04	19.12	16.32
ONS median	6.36	9.00	12.26	15.08	15.89	14.39	12.17
Mod mean	9.05	11.15	14.87	17.35	17.96	17.47	15.41
Mod median	8.20	9.51	12.86	14.41	14.78	14.43	12.56

The Pearson correlations for mean and median between the modelled and raw values are 0.99 and 0.99.

11.4 Commuting flows

11.4.1 List of all workplaces in GB

In order to distribute each individual of the population to a unique physical workplace, we first created a population of all individual workplaces in England, based on a combination of the Nomis UK Business Counts 2020 dataset and the Nomis Business register and Employment Survey 2015 (see Data sources). The first dataset gives the number of individual workplace counts per industry, using the SIC 2007 industry classification, with imprecise size (i.e. number of employees) bands at MSOA level. The second dataset gives the total number of jobs available at LSOA level per SIC 2007 industry category. We found that the distribution of workplace sizes follows closely a simple 1/x distribution, allowing us to draw for each workplace a size

within their band, with sum constraints given by the total number of jobs available, according to the second dataset. The R codes to create the list of all workplaces can be found here.

11.4.2 Usage inside SPC

The workplace 'population' and individual population are levelled for each SIC 2007 category by removing the exceeding part of whichever dataset lists more items. This takes into account that people and business companies are likely to over-report their working availability (e.g. part time and seasonal contracts are not counted differently than full time contracts, job seekers or people on maternity leave might report the SIC of their last job). This process can be controlled by a threshold in the parameter file that defines the maximal total proportion of workers or jobs that can be removed. If the two datasets cannot be levelled accordingly, the categories are dropped and the datasets are levelled globally. Tests in the West Yorkshire area have shown that when the level 1 SIC, containing 21 unique categories, is used, 90% of the volume of commuting flows were recovered compared to the Nomis commuting OD matrices at MSOA level.

The employees for each workplace are drawn according to the 'universal law of visitation', see

> Schläpfer M et al. The universal visitation law of human mobility. Nature 593, 522–527 (2021). (DOI)

This framework predicts that visitors to any destination follow a simple

(r,f)= K / (rf)2

distribution, where (r,f) is the density of visitors coming from a distance r with frequency f and K is a balancing constant depending on the specific area. In the context of commuting, it can be assumed that f = 1. Additionally, we only need to weigh potential employees against each other, which removes the necessity to compute explicitly K. In the West Yorkshire test, we found a Pearson coefficient of 0.7 between the predicted flows when aggregated at MSOA level and the OD matrix at MSOA level available from Nomis.

12 Data schema

12.1 Understanding the schema

Here are some helpful tips for understanding the schema.

Each .pb file contains exactly one `Population` message. In contrast to datasets consisting of multiple `.csv` files, just a single file contains everything. Some of the fields in `Population` are lists (of people and households) or maps (of venues keyed by activity, or of MSOAs). Unlike a flat `.csv` table, there may be more lists embedded later. Each `Household` has a list of `members`, for example.

The different objects refer to each other, forming a graph structure. The protobuf uses `uint64` IDs to index into other lists. For example, if some household has `members = [3, 10]`, then those two people can be found at `population.people[3]` and `population.people[10]`. Each of them will have the same `household` ID, pointing back to something in the `population.households` list.

12.2 Flows: modelling daily activities

SPC models daily travel behaviour of people as "flows." Flows are broken down by an activity – shopping/retail, attending primary or secondary school, working, or staying at home. For each activity type, a person has a list of venues where they may do that activity, weighted by a probability of going to that particular venue.

Note that `flows_per_activity` is stored in `InfoPerMSOA`, not `Person`. The flows for retail and school are only known at the MSOA level, not individually. So given a particular `Person` object, you first look up their household's MSOA – `msoa = population.households[ person.household ].msoa` and then look up flows for that MSOA – `population.info_per_msoa[msoa].flows_per_activity`.

Each person has exactly 1 flow for home – it's just `person.household` with probability 1. A person has 0 or 1 flows to work, based on the value of `person.workplace`.

This doesn't mean that all people in the same MSOA share the same travel behaviour. Each person has their own `activity_durations` field, based on time-use survey data. Even if two

people share the same set of places where they may go shopping, one person may spend much more time on that activity than another.

See the ASPICS conversion script for all of this in action – it has a function to collapse a person's flows down into a single weighted list.

Note that per MSOA, very few venues are represented as destinations – 10 for retail and 5 for school. Only the most likely venues from QUANT are used.

12.3 Flow weights

How do you interpret the probabilities/weights for flows? If your model needs people to visit specific places each day, you could randomly sample a venue from the flows, weighting them appropriately. For retail, you may want to repeat this sampling every day of the simulation, so they visit different venues. For primary and secondary school, it may be more appropriate to sample once and store that for the simulation – a student probably doesn't switch schools daily.

Alternatively, you can follow what ASPICS does. Every day, each person logically visits all possible venues, but their interaction there (possibly receiving or transmitting COVID) is weighted by the probability of each venue.

13 Data sources

The original data are provided at different scales, which define their level of accuracy. For simplicity, the outputs of SPC are geolocated at Output Area (OA) level, although this scale may not be relevant to all indicators. The 2011 OAs are a geographical unit created for census collection and are designed to be relatively homogeneous, with an average size between 120 and 129 households.

The data from Open Street Map (OSM) is downloaded directly from https://www.openstreetmap.org. Everything else is hosted through local copies inside one Azure repository that interacts automatically with the model. We describe below the content of this repository and indicate the raw source used for each indicator. It is divided into utilities, county level data and national data. To recreate the content of this repository from raw sources, please refer to this part of the code.

13.1 Utility data

lookUp-GB.csv.gz

The look-up table links different geographies of Great Britain together. It is used internally by the model, but can also help the user define their own study area. The following are standard denominations, compatible with ONS fields of the same name. They are based on ONS lookups. See ONS documentation for more details.

- `OA11CD`: Output area codes for the 2011 census (120 to 129 households)
- `LSOA11CD` & `LSOA11NM`: Lower-layer Super Output Areas (about 2000 individuals), replaced by Intermediary Zones for Scotland
- `MSOA11CD`,`MSOA11NM`: Middle-layer Super Output Areas (about 8000 individuals), replaced by Data Zones for Scotland
- `LAD20CD`, `LAD20NM`: Local Authority Districts (314 for England, 22 for Wales and 32 for Scotland)
- `ITL321CD`, `ITL321NM`, `ITL221CD`, `ITL221NM`, `ITL121CD` & `ITL121NM`: International Territorial Level, replacing pre-Brexit NUTS European divisions.
- `RGN20CD` & `RGN20NM`: Regions of England (NA for other Wales and Scotland)
- `Country`: England, Wales or Scotland

In addition,

- `AzureRef`: Name of the geographical unit for the County level data folder inside Azure (Lieutenancy Areas – a.k.a. Ceremonial Counties – for England, Scottish Police Divisions and ITL321NM for Wales) For Wales: ITL321NM
- `GoogleMob` & `OSM` are alternate spellings used by Google and OSM for their data releases.

13.2 County level data

Files in this section are grouped by country (England, Wales and Scotland), then date (2012, 2020, 2022, 2032, 2039). The format of a path to an individual file is:

`https://ramp0storage.blob.core.windows.net/countydata-v2-1/[country]/[date]/pop_[area_name].`

where [country], [date] and [area_name] must be replaced accordingly. As of July 2023, England contains 5 series of 47 files, Wales 5 series of 12 files and Scotland 5 series of 13 files

pop_[area_name].csv.gz

The data is mainly based on the 2011 UK census, the UK Time Use Survey 2014-15 and the health surveys of GB (England, Wales, Scotland). The SPENSER microsimulation model is used to distribute and project individuals from the census with MSOA scale constraints into synthetic households with OA constraints. These data are enriched with some of the content of the other datasets mentioned (the rest of which can be added *a posteriori* from the identifiers provided). The data have also been complemented with a modelling of BMI and salaries.

The fields currently contained are detailed in this `.txt` document. They are:

- `pid`: Unique person identifier at GB level within SPC
- `hid`: Unique household identifier at GB level within SPC
- `OA11CD`: Output Area code of the individual's home (ONS, 2011 boundaries)
- `sex`: Sex assigned at birth (DC1117EW, census 2011)
- `age`: Age in years (DC1117EW, census 2011)
- `ethnicity`: Based on self-report (aggregated from DC2101EW, census 2011)
- `nssec8`: National Statistics Socio-economic classification (see methods)
- `HOUSE_nssec8`: National Statistics Socio-economic classification of the reference person of the household (LC4605, census 2011)
- `House_type`: Type of accommodation (based on LC4402EW, census 2011)
- `HOUSE_typeCommunal`: Type of communal establishment (based on QS420, census 2011)
- `HOUSE_NRooms`: Number of rooms in the accommodation (LC4404EW, census 2011)
- `HOUSE_centralHeat`: Presence of central heating (based on LC4402EW, census 2011)

- `HOUSE_tenure`: Tenure (based on LC4402EW, census 2011)
- `HOUSE_NCars`: Number of cars (derived from LC4202EW by SPENSER team, census 2011)
- `id_HS`: unique identifier within the Health Survey (aggregated from the Health surveys from England, Wales and Scotland)
- `HEALTH_diabetes`: for Scotland and England, has doctor diagnosed diabetes; for Wales, diabetes currently treated (derived from HSE, HSW, SHS)
- `HEALTH_bloodpressure`: for Scotland and England, Doctor diagnosed high blood pressure; for Wales, high blood pressure currently treated (derived from HSE, HSW, SHS)
- `HEALTH_cvd`: for England, cardiovascular medication taken in the last 7 days; for Scotland, had cardiovascular condition excluding diabetes / blood pressure; for Wales, any heart condition excluding high blood pressure (derived from HSE, HSW, SHS)
- `HEALTH_NMedicines`: Number of prescribed medications (derived from HSE, HSW, SHS)
- `HEALTH_selfAssessed`: Self assessed general health (derived from HSE, HSW, SHS)
- `HEALTH_lifeSat`: how satisfied with life nowadays? (derived from HSE, HSW, SHS)
- `HEALTH_bmi`: BMI (see methods)
- `id_TUS_hh`: serial household identifier field in the UK Time Use Survey 2015
- `id_TUS_p`: pnum person identifier field in the UK Time Use Survey 2015
- `pwkstat`: Employment status (derived from UK TUS 2015)
- `soc2010`: Standard Occupational Classification (derived from UK TUS 2015)
- `sic1d2007`: Standard Industry Classification of economic activities 2007, 1st level (derived from UK TUS 2015)
- `sic2d2007`: Standard Industry Classification of economic activities 2007, 2nd level (derived from UK TUS 2015)
- `netPayWeekly`: Weekly take home pay after all deductions (derived from UK TUS 2015)
- `workedHoursWeekly`: Number of hours per week usually worked in main job or business (derived from UK TUS 2015)
- `incomeH`: Hourly gross salary for full-time and part-time employees (see methods)
- `incomeY`: Yearly gross salary for full-time and part-time employees (see methods)
- `incomeHAsIf`: Hourly gross salary for employees with self employed/other employees as employees of the same industry and with mean hourly worked for the industry when the number of hours is missing (see methods)
- `incomeYAsIf`: Yearly gross salary for employees with self employed/other employees as employees of the same industry and with mean hourly worked for the industry when the number of hours is missing (see methods)
- `ESport`: Relative probability weight to attend a sport fixture (Experimental, WIP)
- `ERugby`: Relative probability weight to attend a Rugby fixture (Experimental, WIP)
- `EConcertM`: Relative probability weight to attend a concert primarily targeting young males (Experimental, WIP)
- `EConcertF`: Relative probability weight to attend a concert primarily targeting young females (Experimental, WIP)
- `EConcertMS`: Relative probability weight to attend a concert primarily targeting middle-aged males (Experimental, WIP)

- `EConcertMS`: Relative probability weight to attend a concert primarily targeting middle-aged females (Experimental, WIP)
- `EMuseum`: Relative probability weight to visit a museum (Experimental, WIP)
- `easting`: X coordinate of the OA centroid in the British National Grid coordinate system (epsg:27700, source: ONS)
- `northing`: Y coordinate of the OA centroid in the British National Grid coordinate system (epsg:27700, source: ONS)
- `lng`: X coordinate of the OA centroid in the Longitude/Latitude coordinate system (epsg:4326, derived from ONS)
- `lat`: Y coordinate of the OA centroid in the Longitude/Latitude coordinate system (epsg:4326, derived from ONS)

13.3 National data

businessRegistry.csv.gz

Contains a breakdown of all business units (i.e. a single workplace) in Great Britain at LSOA scale, estimated by the project contributors from two nomis datasets: UK Business Counts - local units by industry and employment size band 2020 and Business Register and Employment Survey 2015. Each item contains the `size` of the unit and its main `sic1d07` code in reference to standard Industrial Classification of Economic Activities 2007 (number corresponding to the letter in alphabetical order). It is used to compute commuting flows.

`GIS/`

This directory contains three GIS datasets of GB in GeoJson format taken from ONS boundaries:

- OA_2011_Pop20.geojson at OA level
- LSOA_2011_Pop20.geojson at LSOA level
- MSOA_2011_Pop20.geojson at MSOA level

QUANT_RAMP_spc.tar.gz

See: Milton R, Batty M, Dennett A, dedicated RAMP Spatial Interaction Model GitHub repository. It is used to compute the flows towards schools and retail.

timeAtHomeIncreaseCTY.csv.gz

This file is a subset from Google COVID-19 Community Mobility Reports, cropped to GB. It describes the daily reduction in mobility, averaged at county level, due to lockdown and other COVID-19 restrictions between the 15th of February 2020 and 15th of October 2022. Missing values have been replaced by the national average. These values can be used directly to reduce `pnothome` and increase `phometot` (and their sub-categories) to simulate more accurately the period.

diariesRef.csv.gz

Contains diaries taken from the UK TUS that can be distributed to the population on a daily basis. They contain weekend days and weekday days. A full description of the fields can be found here.

Validation

Currently, the validation of each element of the methods is mentioned (or referenced) inside the modelling methods section.

A sytematic validation of the model is under construction. Approximative ETA: September 2023.

Part III

Advanced

14 Developer guide

14.1 Updating the docs

The site is built with Quarto. You can iterate on it locally: `cd docs; quarto preview`

14.2 Code hygiene

We use automated tools to format the code.

```
cargo fmt

# Format Markdown docs
prettier --write *.md
prettier --write docs/*.qmd --parser markdown
```

Install prettier for Markdown.

14.3 Some tips for working with Rust

There are two equivalent ways to rebuild and then run the code. First:

```
cargo run --release -- devon
```

The `--` separates arguments to `cargo`, the Rust build tool, and arguments to the program itself. The second way:

```
cargo build --release
./target/release/aspics devon
```

You can build the code in two ways – **debug** and **release**. There's a simple tradeoff – debug mode is fast to build, but slow to run. Release mode is slow to build, but fast to run. For the ASPICS codebase, since the input data is so large and the codebase so small, I'd recommend always using `--release`. If you want to use debug mode, just omit the flag.

If you're working on the Rust code outside of an IDE like VSCode, then you can check if the code compiles much faster by doing `cargo check`.

14.4 Docker

We provide a Dockerfile in case it's helpful for running, but don't recommend using it. If you want to, then assuming you have Docker setup:

```
docker build -t spc .
docker run --mount type=bind,source="$(pwd)"/data,target=/spc/data -t spc /spc/target/release
```

This will make the `data` directory in your directory available to the Docker image, where it'll download the large input files and produce the final output.

15 Code walkthrough

SPC is implemented in Rust, and its code can be found here. This is an unusual implementation choice in the data science world, so this page has some notes about it.

15.1 Generally useful techniques

The code-base makes use of some techniques that may be generally applicable to other projects, independent of the language chosen.

15.1.1 Split code into two stages

Agent-based models and spatial interaction models require some kind of input. Often the effort to transform external data into this input can exceed that of the simulation component. Cleanly separating the two problems has some advantages:

- iterate on the simulation faster, without processing raw data every run
- reuse the prepared input for future projects
- force thinking about the data model needed by the simulation, and transform the external data into that form

SPC is exactly this first stage, originally split from ASPICS when further uses of the same population data were identified.

15.1.2 Explicit data schema

Dynamically typed languages like Python don't force you to explicitly list the shape of input data. It's common to read CSV files with `pandas`, filter and transform the data, and use that throughout the program. This can be quick to start prototyping, but is hard to maintain longer-term. Investing in the process of writing down types:

- makes it easier for somebody new to understand your system – they can first focus on **what** you're modeling, instead of how that's built up from raw data sources
- clarifies what data actually matters to your system; you don't carry forward unnecessary input

- makes it impossible to express invalid states
  - One example is here – per person and activity, there's a list of venues the person may visit, along with a probability of going there. If the list of venues and list of probabilities are stored as separate lists or columns, then their length may not match.
- reuse the prepared input for future projects

There's a variety of techniques for expressing strongly typed data:

- protocol buffers or flatbuffers
- JSON schemas
- Python data classes and optional type hints
- statically typed languages like Rust

15.1.3 Type-safe IDs

Say your data model has many different objects, each with their own ID – people, households, venues, etc. You might store these in a list and use the index as an ID. This is fine, but nothing stops you from confusing IDs and accidentally passing in venue 5 to a function instead of household 5. In Rust, it's easy to create "wrapper types" like this and let the compiler prevent these mistakes.

This technique is also useful when preparing external data. GTFS data describing public transit routes and timetables contains many string IDs – shapes, trips, stops, routes. As soon as you read the raw input, you can store the strings in more precise types that prevent mixing up a stop ID and route ID.

15.1.4 Idempotent data preparation

If you're iterating on your initialisation pipeline's code, you probably don't want to download a 2GB external file every single run. A common approach is to first test if a file exists and don't download it again if so. In practice, you may also need to handle unzipping files, showing a progress bar while downloading, and printing clear error messages. This codebase has some common code for doing this in Rust. We intend to publish a separate library to more easily call in your own code.

15.1.5 Logging with structure

It's typical to print information as a complex pipeline runs, for the user to track progress and debug problems. But without any sort of organization, it's hard to follow what steps take a long time or encounter problems. What if your logs could show the logical structure of your pipeline and help you understand where time is spent?

```
[192.30s] [get_info_per_msoa] Loading buildings from data/raw_data/countydata/OSM/west-yorkshire-latest-free/
[192.64s] [get_info_per_msoa] Found 474,207 buildings from data/raw_data/countydata/OSM/west-yorkshire-latest-free/gis
osm_buildings_a_free_1.shp
[192.70s] [get_info_per_msoa] Matching 474,207 points to 299 polygons. Building R-Tree...
[194.22s] [calculate_lockdown_per_day] Calculating per-day lockdown values
[194.24s] [load_events] Loading events data
[194.25s] [initialisation] By the end, Memory usage: 1.53GiB
[200.89s] [Writing snapshot] Merging flows for all activities

  212.24s          initialisation WestYorkshireLarge
    31.18ms          grab_raw_data
    192.04s          creating population
      8.20s            read_individual_time_use_and_health_data
        4.35s            Reading "data/raw_data/countydata/tus_hse_west-yorkshire.csv"
        3.83s            Creating households
      152.38s          create_commuting_flows
      8.30s            setup_venue_flows Retail
        6.59s            Copying flows to people Retail
      7.47s            setup_venue_flows Nightclub
        6.63s            Copying flows to people Nightclub
      8.68s            setup_venue_flows PrimarySchool
        6.58s            Copying flows to people PrimarySchool
      7.00s            setup_venue_flows SecondarySchool
        6.50s            Copying flows to people SecondarySchool
    2.03s            get_info_per_msoa
    24.48ms          calculate_lockdown_per_day
    251.20µs         load_events "model_parameters/eventDataConcerts.csv"
    1.07s            Writing population to "data/processed_data/WestYorkshireLarge/rust_cache.bin"
    16.93s           Writing snapshot
```

The screenshot above shows a summary printed at the end of a long pipeline run. It's immediately obvious that the slowest step is creating commuting flows.

This codebase uses the tracing framework for logging, with a custom piece to draw the tree. (We'll publish this as a separate library once it's more polished.) The tracing framework is hard to understand, but the main conceptual leap over regular logging framworks is the concept of a **span**. When your code starts one logical step, you call a method to create a new span, and when it finishes, you close that span. Spans can be nested in any way – `create_commuting_flows` happens within the larger step of `creating population`.

15.1.6 Determinism

Given the same inputs, your code should always produce identical output, no matter where it's run or how many times. Otherwise, debugging problems becomes very tedious, and it's more difficult to make conclusions from results. Of course, many projects have a stochastic element – but this should be controlled by a random number generator (RNG) seed, which is part of the input. You vary the seed and repeat the program, then reason about the distribution of results.

Aside from organizing your code to let a single RNG seed influence everything, another possible source of non-determinism is iteration order. In Rust, a `HashMap` could have different order every time it's used, so we use a `BTreeMap` instead when this matters. In Python, dictionaries are ordered. Be sure to check for your language.

15.2 Protocol buffers

SPC uses protocol buffers v2 for output. This has some advantages explained the "explicit data schema" section above.

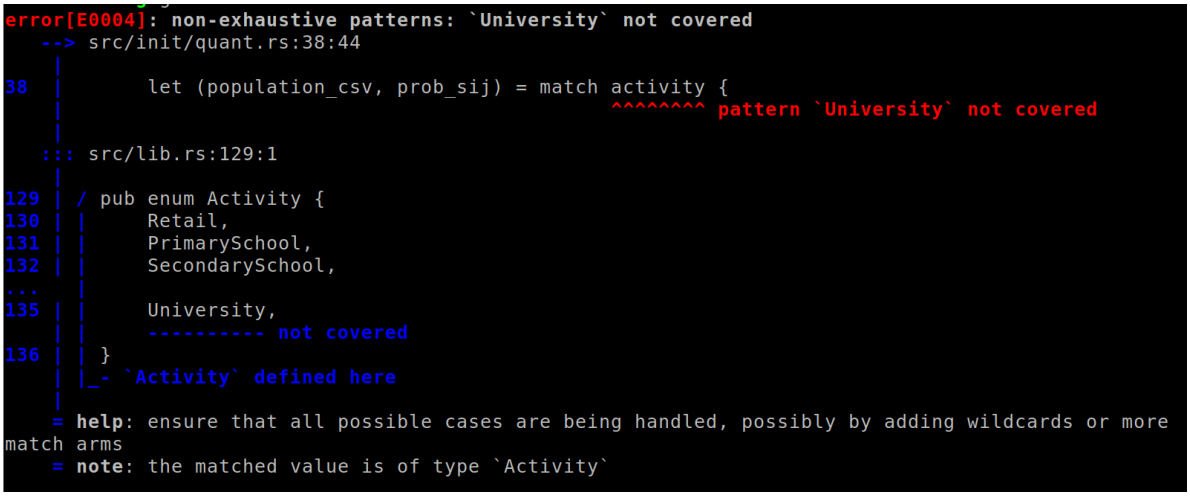
Note that we chose proto2 instead of proto3, because proto3 doesn't support required fields. This is done to allow schemas to evolve better over time, but this isn't a feature SPC makes use of. There's no need to have new code work with old data, or vice versa – if the schema is updated, downstream code should adapt accordingly and use the updated input files.

Note also that protocol buffers don't easily support type-safe wrappers around numeric IDs, so downstream code has to be careful not to mix up household, venue, and person IDs. For this reason, SPC internally doesn't use the auto-generated protobuf code until the very end of the pipeline. It's always possible to be more precise with native Rust types, and convert to the less strict types later.

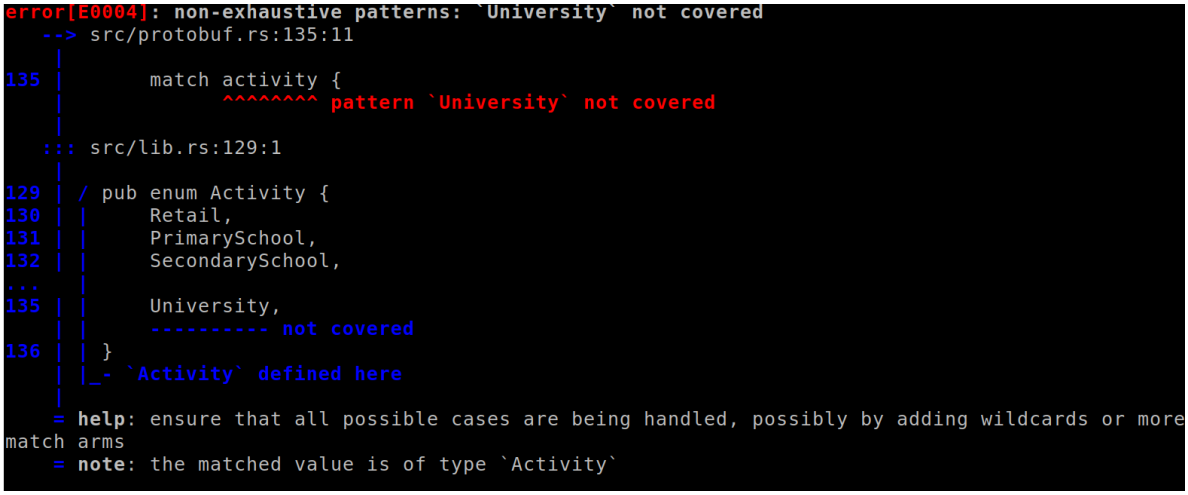
15.3 An example of the power of static type checking

Imagine we want to add a new activity type to represent people going to university and higher education. SPC already has activities for primary and secondary school, so we'll probably want to follow those as a guide. In any language, we could search the codebase for relevant terms to get a sense of what to update. In languages like Python without an up-front compilation step, if we fail to update something or write blatantly incorrect code (such as making a typo in variable names or passing a list where a string was expected), we only find out when that code happens to run. In pipelines with many steps and large input files, it could be a while before we reach the problematic code.

Let's walk through the same exercise for SPC's Rust code. We start by adding a new `University` case to the Activity enum. If we try to compile the code here (with `cargo check` or an IDE), we immediately get 4 errors.



Three of the errors are in the QUANT module. The first is here. It's immediately clear that for retail and primary/secondary school, we read in two files from QUANT representing venues where these activities take place and the probability of going to each venue. Even if we were unfamiliar with this codebase, the compiler has told us one thing we'll need to figure out, and where to wire it up.



The other error is in the code that writes the protobuf output. Similarly, we need a way to represent university activities in the protobuf scheme.

Extending an unfamiliar code-base backed by compiler errors is a very guided experience. If you wanted to add more demographic attributes to people or energy use information to households, you don't need to guess all of the places in the code you'll need to update. You can just add the field, then let the compiler tell you all places where those objects get created.

16 Performance

The following table summarizes the resources SPC needs to run in different areas.

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Runtime (Commute)	Memory_use
2012	England/bedfordshire	74	245,166	647,272	256.91 MiB	7 sec	2 sec	848.99 MiB
2020	England/bedfordshire	74	272,875	674,044	271.73 MiB	7 sec	2 sec	922.86 MiB
2022	England/bedfordshire	74	309,706	703,582	277.82 MiB	7 sec	2 sec	929.78 MiB
2032	England/bedfordshire	74	309,706	703,582	277.82 MiB	7 sec	2 sec	929.78 MiB
2039	England/bedfordshire	74	329,061	715,797	278.47 MiB	7 sec	2 sec	927.74 MiB
2012	England/berkshire	107	342,167	890,543	356.08 MiB	10 sec	4 sec	1.06 GiB
2020	England/berkshire	107	365,905	918,258	373.39 MiB	10 sec	4 sec	1.10 GiB
2022	England/berkshire	107	394,446	941,655	368.41 MiB	10 sec	4 sec	1.08 GiB
2032	England/berkshire	107	394,446	941,655	368.41 MiB	10 sec	4 sec	1.08 GiB
2039	England/berkshire	107	408,604	949,986	367.25 MiB	10 sec	4 sec	1.07 GiB
2012	England/bristol	55	182,299	448,233	173.75 MiB	5 sec	1 sec	527.15 MiB
2020	England/bristol	55	196,940	470,039	184.00 MiB	5 sec	1 sec	547.40 MiB
2022	England/bristol	55	216,197	503,014	192.51 MiB	5 sec	1 sec	559.70 MiB
2032	England/bristol	55	216,197	503,014	192.51 MiB	6 sec	1 sec	559.70 MiB
2039	England/bristol	55	227,770	521,371	199.73 MiB	6 sec	1 sec	573.32 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Runtime	(Commute)	Memory_use
2012	England/buckinghamshire	99	301,486	786,221	314.40 MiB	9 sec	3 sec	1007.27 MiB
2020	England/buckinghamshire	99	327,554	816,518	331.16 MiB	9 sec	3 sec	1.02 GiB
2022	England/buckinghamshire	99	333,801	824,863	334.87 MiB	9 sec	3 sec	1.03 GiB
2032	England/buckinghamshire	99	363,840	844,684	331.67 MiB	9 sec	3 sec	1.01 GiB
2039	England/buckinghamshire	99	381,583	855,739	332.20 MiB	9 sec	3 sec	1.01 GiB
2012	England/cambridgeshire	98	327,257	832,980	323.39 MiB	9 sec	3 sec	1013.07 MiB
2020	England/cambridgeshire	98	348,522	863,250	341.20 MiB	9 sec	3 sec	1.03 GiB
2022	England/cambridgeshire	98	377,634	907,166	348.79 MiB	9 sec	3 sec	1.03 GiB
2032	England/cambridgeshire	98	377,634	907,166	348.79 MiB	9 sec	3 sec	1.03 GiB
2039	England/cambridgeshire	98	392,478	924,170	351.43 MiB	9 sec	3 sec	1.04 GiB
2012	England/cheshire	139	441,084	1,042,065	402.31 MiB	12 sec	4 sec	1.13 GiB
2020	England/cheshire	139	464,134	1,070,597	416.52 MiB	12 sec	4 sec	1.46 GiB
2022	England/cheshire	139	489,476	1,125,198	425.44 MiB	12 sec	4 sec	1.47 GiB
2032	England/cheshire	139	489,476	1,125,198	425.44 MiB	12 sec	4 sec	1.47 GiB
2039	England/cheshire	139	501,501	1,149,515	431.28 MiB	12 sec	4 sec	1.48 GiB
2012	England/cornwall	74	233,710	551,951	208.93 MiB	7 sec	2 sec	744.32 MiB
2020	England/cornwall	74	248,145	579,460	220.51 MiB	7 sec	2 sec	766.20 MiB
2022	England/cornwall	74	251,934	590,365	224.28 MiB	7 sec	2 sec	773.13 MiB
2032	England/cornwall	74	271,147	636,573	234.01 MiB	7 sec	2 sec	829.51 MiB
2039	England/cornwall	74	281,563	660,164	240.35 MiB	7 sec	2 sec	839.16 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Runtime	(Commute)	Memory_use
2012	England/cumbria	64	222,586	498,624	188.07 MiB	6 sec	1 sec	547.25 MiB
2020	England/cumbria	64	226,893	499,873	188.76 MiB	6 sec	1 sec	548.43 MiB
2022	England/cumbria	64	230,206	499,840	183.22 MiB	6 sec	1 sec	533.91 MiB
2032	England/cumbria	64	230,206	499,840	183.22 MiB	6 sec	1 sec	533.91 MiB
2039	England/cumbria	64	231,202	498,475	181.62 MiB	6 sec	1 sec	530.88 MiB
2012	England/derbyshire	131	436,276	1,035,356	397.76 MiB	11 sec	4 sec	1.12 GiB
2020	England/derbyshire	131	459,743	1,064,406	409.77 MiB	11 sec	4 sec	1.44 GiB
2022	England/derbyshire	131	489,764	1,122,078	419.53 MiB	12 sec	4 sec	1.45 GiB
2032	England/derbyshire	131	489,764	1,122,078	419.53 MiB	12 sec	4 sec	1.45 GiB
2039	England/derbyshire	131	505,314	1,152,518	429.02 MiB	12 sec	4 sec	1.47 GiB
2012	England/devon	156	494,106	1,165,952	438.76 MiB	13 sec	4 sec	1.49 GiB
2020	England/devon	156	523,033	1,212,387	459.60 MiB	13 sec	4 sec	1.53 GiB
2022	England/devon	156	567,011	1,304,874	478.87 MiB	14 sec	4 sec	1.64 GiB
2032	England/devon	156	567,011	1,304,874	478.87 MiB	14 sec	5 sec	1.64 GiB
2039	England/devon	156	589,178	1,342,775	488.39 MiB	14 sec	5 sec	1.66 GiB
2012	England/dorset	95	328,906	761,766	285.99 MiB	8 sec	2 sec	931.64 MiB
2020	England/dorset	95	345,862	777,887	295.20 MiB	8 sec	2 sec	951.30 MiB
2022	England/dorset	95	350,392	782,725	296.83 MiB	8 sec	2 sec	955.86 MiB
2032	England/dorset	95	375,160	802,953	294.92 MiB	8 sec	2 sec	945.43 MiB
2039	England/dorset	95	389,694	810,856	294.90 MiB	8 sec	2 sec	945.59 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Pa	ntim	(C	ommute)	Memory_use
2012	England/durham	117	390,472	911,601	349.81 MiB	9 sec		3 sec		1.03 GiB
2020	England/durham	117	407,828	930,184	359.62 MiB	9 sec		3 sec		1.05 GiB
2022	England/durham	117	425,611	952,801	356.65 MiB	9 sec		3 sec		1.03 GiB
2032	England/durham	117	425,611	952,801	356.65 MiB	9 sec		3 sec		1.03 GiB
2039	England/durham	117	434,593	959,555	357.69 MiB	9 sec		3 sec		1.04 GiB
2012	England/east-sussex	102	355,257	827,703	313.77 MiB	9 sec		3 sec		987.24 MiB
2020	England/east-sussex	102	380,894	853,970	324.07 MiB	9 sec		3 sec		1006.06 MiB
2022	England/east-sussex	102	423,181	895,907	329.61 MiB	9 sec		3 sec		1008.52 MiB
2032	England/east-sussex	102	423,181	895,907	329.61 MiB	9 sec		3 sec		1008.52 MiB
2039	England/east-sussex	102	446,000	915,014	335.50 MiB	9 sec		3 sec		1020.68 MiB
2012	England/east-yorkshire-with-hull	75	255,848	593,271	227.51 MiB	7 sec		2 sec		778.67 MiB
2020	England/east-yorkshire-with-hull	75	262,609	602,286	233.16 MiB	7 sec		2 sec		834.96 MiB
2022	England/east-yorkshire-with-hull	75	272,805	613,721	230.36 MiB	7 sec		2 sec		824.41 MiB
2032	England/east-yorkshire-with-hull	75	272,805	613,721	230.36 MiB	7 sec		2 sec		824.42 MiB
2039	England/east-yorkshire-with-hull	75	277,770	617,357	230.47 MiB	7 sec		2 sec		824.92 MiB
2012	England/essex	211	722,974	1,786,310	690.86 MiB	19 sec		9 sec		2.06 GiB
2020	England/essex	211	773,454	1,857,205	726.11 MiB	20 sec		9 sec		2.13 GiB
2022	England/essex	211	858,552	1,981,994	761.49 MiB	21 sec		9 sec		2.19 GiB
2032	England/essex	211	858,552	1,981,994	761.49 MiB	21 sec		10 sec		2.19 GiB
2039	England/essex	211	906,640	2,042,404	777.80 MiB	22 sec		10 sec		2.21 GiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Runtime	(Commute)	Memory_use
2012	England/gloucestershire	107	365,240	889,836	344.21 MiB	10 sec	3 sec	1.02 GiB
2020	England/gloucestershire	107	392,643	933,909	362.94 MiB	11 sec	3 sec	1.06 GiB
2022	England/gloucestershire	107	432,216	1,025,077	389.60 MiB	11 sec	3 sec	1.10 GiB
2032	England/gloucestershire	107	432,216	1,025,077	389.60 MiB	11 sec	3 sec	1.10 GiB
2039	England/gloucestershire	107	453,383	1,068,484	403.92 MiB	11 sec	3 sec	1.43 GiB
2012	England/greater-london	983	3,287,651	8,587,955	3.28 GiB	5 min	4 min	11.80 GiB
2020	England/greater-london	983	3,578,616	8,992,494	3.48 GiB	5 min	4 min	12.22 GiB
2022	England/greater-london	983	3,645,459	9,105,919	3.53 GiB	5 min	4 min	12.31 GiB
2032	England/greater-london	983	4,001,897	9,461,273	3.55 GiB	5 min	5 min	12.26 GiB
2039	England/greater-london	983	4,233,367	9,697,960	3.59 GiB	6 min	5 min	12.96 GiB
2012	England/greater-manchester	346	1,128,371	2,745,455	1.05 GiB	40 sec	26 sec	3.56 GiB
2020	England/greater-manchester	346	1,192,547	2,840,431	1.10 GiB	41 sec	27 sec	3.66 GiB
2022	England/greater-manchester	346	1,272,689	2,974,954	1.13 GiB	43 sec	27 sec	3.69 GiB
2032	England/greater-manchester	346	1,272,689	2,974,954	1.13 GiB	43 sec	28 sec	3.69 GiB
2039	England/greater-manchester	346	1,319,090	3,049,727	1.15 GiB	45 sec	29 sec	3.73 GiB
2012	England/hampshire	225	733,611	1,810,518	698.19 MiB	21 sec	10 sec	2.07 GiB
2020	England/hampshire	225	777,116	1,861,250	721.78 MiB	21 sec	10 sec	2.12 GiB
2022	England/hampshire	225	836,451	1,931,669	729.13 MiB	21 sec	10 sec	2.12 GiB
2032	England/hampshire	225	836,451	1,931,669	729.13 MiB	21 sec	10 sec	2.12 GiB
2039	England/hampshire	225	867,417	1,960,190	735.66 MiB	22 sec	10 sec	2.13 GiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Runtime	(Commute)	Memory_use
2012	England/herefordshire	23	79,083	188,362	72.22 MiB	3 sec	1 sec	234.79 MiB
2020	England/herefordshire	23	83,238	195,194	74.72 MiB	3 sec	1 sec	239.26 MiB
2022	England/herefordshire	23	89,574	209,784	77.64 MiB	3 sec	1 sec	242.72 MiB
2032	England/herefordshire	23	89,574	209,784	77.64 MiB	3 sec	1 sec	242.72 MiB
2039	England/herefordshire	23	92,605	216,508	79.44 MiB	3 sec	1 sec	245.59 MiB
2012	England/hertfordshire	153	457,276	1,160,155	458.74 MiB	13 sec	5 sec	1.56 GiB
2020	England/hertfordshire	153	494,661	1,190,043	477.27 MiB	13 sec	5 sec	1.59 GiB
2022	England/hertfordshire	153	546,573	1,219,124	476.65 MiB	13 sec	5 sec	1.67 GiB
2032	England/hertfordshire	153	546,573	1,219,124	476.65 MiB	13 sec	5 sec	1.67 GiB
2039	England/hertfordshire	153	575,179	1,233,573	477.07 MiB	13 sec	5 sec	1.67 GiB
2012	England/isle-of-wight	18	61,636	139,732	53.88 MiB	3 sec	1 sec	188.67 MiB
2020	England/isle-of-wight	18	65,140	143,268	54.99 MiB	3 sec	1 sec	190.34 MiB
2022	England/isle-of-wight	18	70,496	151,582	55.55 MiB	3 sec	1 sec	200.88 MiB
2032	England/isle-of-wight	18	70,496	151,582	55.55 MiB	3 sec	1 sec	200.88 MiB
2039	England/isle-of-wight	18	72,968	154,841	56.14 MiB	3 sec	1 sec	202.02 MiB
2012	England/kent	220	718,544	1,793,702	700.26 MiB	19 sec	8 sec	2.08 GiB
2020	England/kent	220	781,933	1,873,451	737.36 MiB	20 sec	9 sec	2.15 GiB
2022	England/kent	220	875,515	2,008,857	773.40 MiB	20 sec	9 sec	2.21 GiB
2032	England/kent	220	875,515	2,008,857	773.40 MiB	20 sec	9 sec	2.21 GiB
2039	England/kent	220	926,571	2,069,087	788.63 MiB	21 sec	9 sec	2.23 GiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Rayon_time	(Commute)	Memory_use
2012	England/lancashire	191	619,861	1,476,469	572.04 MiB	16 sec	7 sec	1.83 GiB
2020	England/lancashire	191	640,196	1,511,896	589.88 MiB	16 sec	7 sec	1.87 GiB
2022	England/lancashire	191	663,637	1,567,390	594.59 MiB	16 sec	7 sec	1.87 GiB
2032	England/lancashire	191	663,637	1,567,390	594.59 MiB	16 sec	7 sec	1.87 GiB
2039	England/lancashire	191	674,387	1,591,908	600.12 MiB	17 sec	7 sec	1.88 GiB
2012	England/leicestershire	120	391,605	1,014,485	394.46 MiB	10 sec	4 sec	1.12 GiB
2020	England/leicestershire	120	418,618	1,073,842	419.67 MiB	11 sec	4 sec	1.47 GiB
2022	England/leicestershire	120	424,923	1,092,677	426.66 MiB	11 sec	4 sec	1.49 GiB
2032	England/leicestershire	120	460,335	1,178,746	449.47 MiB	12 sec	5 sec	1.52 GiB
2039	England/leicestershire	120	482,373	1,225,824	464.68 MiB	12 sec	4 sec	1.55 GiB
2012	England/lincolnshire	134	449,394	1,064,403	403.11 MiB	11 sec	4 sec	1.43 GiB
2020	England/lincolnshire	134	475,646	1,098,403	419.38 MiB	11 sec	4 sec	1.46 GiB
2022	England/lincolnshire	134	507,295	1,152,299	427.62 MiB	11 sec	4 sec	1.47 GiB
2032	England/lincolnshire	134	507,295	1,152,299	427.62 MiB	11 sec	4 sec	1.47 GiB
2039	England/lincolnshire	134	523,548	1,172,923	430.89 MiB	11 sec	4 sec	1.47 GiB
2012	England/merseyside	184	603,483	1,399,209	533.99 MiB	14 sec	6 sec	1.75 GiB
2020	England/merseyside	184	632,617	1,435,755	553.36 MiB	14 sec	6 sec	1.79 GiB
2022	England/merseyside	184	665,766	1,498,518	570.24 MiB	14 sec	6 sec	1.82 GiB
2032	England/merseyside	184	665,766	1,498,518	570.24 MiB	14 sec	6 sec	1.82 GiB
2039	England/merseyside	184	685,165	1,528,037	577.51 MiB	15 sec	6 sec	1.83 GiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Rayon time	(Commute)	Memory_use
2012	England/norfolk	110	374,491	882,793	333.12 MiB	10 sec	3 sec	1017.08 MiB
2020	England/norfolk	110	397,770	916,799	348.46 MiB	10 sec	3 sec	1.02 GiB
2022	England/norfolk	110	432,187	982,755	362.33 MiB	10 sec	3 sec	1.04 GiB
2032	England/norfolk	110	432,187	982,755	362.33 MiB	10 sec	3 sec	1.04 GiB
2039	England/norfolk	110	450,068	1,013,214	371.44 MiB	10 sec	3 sec	1.06 GiB
2012	England/north-yorkshire	138	460,050	1,085,067	413.12 MiB	12 sec	4 sec	1.45 GiB
2020	England/north-yorkshire	138	478,639	1,107,928	423.25 MiB	12 sec	4 sec	1.47 GiB
2022	England/north-yorkshire	138	499,392	1,134,723	420.66 MiB	12 sec	4 sec	1.45 GiB
2032	England/north-yorkshire	138	499,392	1,134,723	420.66 MiB	12 sec	4 sec	1.45 GiB
2039	England/north-yorkshire	138	509,099	1,143,895	421.58 MiB	12 sec	4 sec	1.46 GiB
2012	England/northamptonshire	91	289,575	720,263	284.41 MiB	8 sec	2 sec	941.24 MiB
2020	England/northamptonshire	91	316,553	762,382	304.38 MiB	8 sec	2 sec	981.06 MiB
2022	England/northamptonshire	91	352,529	828,003	320.83 MiB	9 sec	3 sec	1005.56 MiB
2032	England/northamptonshire	91	352,529	828,003	320.83 MiB	9 sec	3 sec	1005.56 MiB
2039	England/northamptonshire	91	370,555	855,812	328.05 MiB	9 sec	3 sec	1016.77 MiB
2012	England/northumberland	40	138,928	315,894	120.67 MiB	5 sec	1 sec	423.02 MiB
2020	England/northumberland	40	143,516	322,616	121.95 MiB	5 sec	1 sec	423.78 MiB
2022	England/northumberland	40	148,792	333,456	122.08 MiB	5 sec	1 sec	421.39 MiB
2032	England/northumberland	40	148,792	333,456	122.08 MiB	5 sec	1 sec	421.39 MiB
2039	England/northumberland	40	150,259	337,186	122.26 MiB	5 sec	1 sec	421.38 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Runtime	(Commute)	Memory_use
2012	England/nottinghamshire	138	460,022	1,123,005	432.55 MiB	12 sec	4 sec	1.49 GiB
2020	England/nottinghamshire	138	486,163	1,169,489	453.88 MiB	12 sec	4 sec	1.53 GiB
2022	England/nottinghamshire	138	522,944	1,248,804	473.55 MiB	12 sec	5 sec	1.56 GiB
2032	England/nottinghamshire	138	522,944	1,248,804	473.55 MiB	12 sec	5 sec	1.56 GiB
2039	England/nottinghamshire	138	543,291	1,281,812	482.41 MiB	13 sec	5 sec	1.66 GiB
2012	England/oxfordshire	86	261,235	671,997	260.47 MiB	7 sec	2 sec	852.78 MiB
2020	England/oxfordshire	86	274,908	695,490	271.66 MiB	7 sec	2 sec	918.84 MiB
2022	England/oxfordshire	86	293,368	729,866	275.44 MiB	7 sec	2 sec	919.28 MiB
2032	England/oxfordshire	86	293,368	729,866	275.44 MiB	8 sec	2 sec	919.28 MiB
2039	England/oxfordshire	86	303,035	743,227	277.55 MiB	8 sec	2 sec	922.13 MiB
2012	England/rutland	5	14,912	38,314	16.37 MiB	2 sec	1 sec	53.95 MiB
2020	England/rutland	5	16,698	40,381	17.09 MiB	2 sec	1 sec	57.84 MiB
2022	England/rutland	5	18,198	44,193	18.26 MiB	2 sec	1 sec	59.97 MiB
2032	England/rutland	5	18,198	44,193	18.26 MiB	2 sec	1 sec	59.97 MiB
2039	England/rutland	5	18,914	45,659	18.71 MiB	2 sec	1 sec	61.09 MiB
2012	England/shropshire	62	197,768	483,414	186.37 MiB	6 sec	1 sec	550.90 MiB
2020	England/shropshire	62	211,035	508,233	195.85 MiB	6 sec	1 sec	568.56 MiB
2022	England/shropshire	62	228,285	558,755	207.37 MiB	6 sec	1 sec	740.52 MiB
2032	England/shropshire	62	228,285	558,755	207.37 MiB	6 sec	1 sec	740.52 MiB
2039	England/shropshire	62	236,015	581,476	213.31 MiB	6 sec	1 sec	749.75 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Rayon	(Commute)	Memory_use
2012	England/somerset	124	392,224	938,968	359.26 MiB	10 sec	3 sec	1.05 GiB
2020	England/somerset	124	421,693	979,526	376.56 MiB	10 sec	3 sec	1.08 GiB
2022	England/somerset	124	428,543	993,364	381.41 MiB	10 sec	3 sec	1.09 GiB
2032	England/somerset	124	463,526	1,054,161	394.38 MiB	11 sec	3 sec	1.41 GiB
2039	England/somerset	124	484,587	1,087,596	404.50 MiB	11 sec	3 sec	1.43 GiB
2012	England/south-yorkshire	172	566,664	1,372,435	528.13 MiB	14 sec	6 sec	1.75 GiB
2020	England/south-yorkshire	172	597,694	1,418,840	548.61 MiB	15 sec	6 sec	1.79 GiB
2022	England/south-yorkshire	172	637,411	1,493,544	563.93 MiB	15 sec	6 sec	1.81 GiB
2032	England/south-yorkshire	172	637,411	1,493,544	563.93 MiB	15 sec	6 sec	1.81 GiB
2039	England/south-yorkshire	172	659,843	1,531,313	575.33 MiB	15 sec	6 sec	1.83 GiB
2012	England/staffordshire	143	464,441	1,111,144	425.33 MiB	12 sec	4 sec	1.47 GiB
2020	England/staffordshire	143	486,645	1,139,752	437.56 MiB	12 sec	4 sec	1.49 GiB
2022	England/staffordshire	143	510,634	1,188,857	444.92 MiB	12 sec	4 sec	1.50 GiB
2032	England/staffordshire	143	510,634	1,188,857	444.92 MiB	12 sec	4 sec	1.50 GiB
2039	England/staffordshire	143	522,882	1,215,006	453.00 MiB	12 sec	4 sec	1.52 GiB
2012	England/suffolk	90	312,178	746,863	285.39 MiB	8 sec	2 sec	933.65 MiB
2020	England/suffolk	90	331,778	766,023	294.07 MiB	8 sec	2 sec	950.73 MiB
2022	England/suffolk	90	336,599	773,019	296.48 MiB	8 sec	2 sec	956.16 MiB
2032	England/suffolk	90	360,555	800,189	298.09 MiB	8 sec	2 sec	952.75 MiB
2039	England/suffolk	90	375,536	817,179	302.95 MiB	8 sec	2 sec	963.06 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Runtime	(Commute)	Memory_use
2012	England/surrey	151	458,108	1,168,112	456.56 MiB	14 sec	7 sec	1.55 GiB
2020	England/surrey	151	480,930	1,195,509	472.95 MiB	14 sec	6 sec	1.58 GiB
2022	England/surrey	151	518,720	1,214,557	467.08 MiB	14 sec	6 sec	1.56 GiB
2032	England/surrey	151	518,720	1,214,557	467.08 MiB	14 sec	6 sec	1.56 GiB
2039	England/surrey	151	538,941	1,221,227	464.76 MiB	14 sec	6 sec	1.64 GiB
2012	England/tyne-and-wear	145	483,909	1,119,030	427.37 MiB	11 sec	4 sec	1.47 GiB
2020	England/tyne-and-wear	145	501,383	1,143,194	439.11 MiB	11 sec	4 sec	1.50 GiB
2022	England/tyne-and-wear	145	521,777	1,168,078	440.06 MiB	11 sec	4 sec	1.49 GiB
2032	England/tyne-and-wear	145	521,777	1,168,078	440.06 MiB	11 sec	4 sec	1.49 GiB
2039	England/tyne-and-wear	145	532,652	1,177,340	441.39 MiB	11 sec	4 sec	1.58 GiB
2012	England/warwickshire	108	361,467	896,673	347.46 MiB	10 sec	3 sec	1.03 GiB
2020	England/warwickshire	108	392,639	958,833	373.64 MiB	10 sec	3 sec	1.08 GiB
2022	England/warwickshire	108	432,682	1,061,955	405.97 MiB	11 sec	4 sec	1.44 GiB
2032	England/warwickshire	108	432,682	1,061,955	405.97 MiB	11 sec	4 sec	1.44 GiB
2039	England/warwickshire	108	454,732	1,112,230	424.11 MiB	11 sec	4 sec	1.47 GiB
2012	England/west-midlands	314	958,034	2,477,391	990.28 MiB	33 sec	19 sec	3.24 GiB
2020	England/west-midlands	314	1,002,273	2,572,395	1.01 GiB	34 sec	19 sec	3.33 GiB
2022	England/west-midlands	314	1,046,146	2,664,228	1.04 GiB	35 sec	20 sec	3.37 GiB
2032	England/west-midlands	314	1,079,612	2,706,242	1.04 GiB	36 sec	21 sec	3.55 GiB
2039	England/west-midlands	314	1,128,890	2,787,990	1.07 GiB	38 sec	22 sec	3.59 GiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Pa_ntime	(C_ommute)	Memory_use
2012	England/west-sussex	100	348,766	836,646	321.38 MiB	9 sec	3 sec	1004.51 MiB
2020	England/west-sussex	100	375,837	871,029	337.97 MiB	9 sec	3 sec	1.01 GiB
2022	England/west-sussex	100	419,347	931,573	350.32 MiB	9 sec	3 sec	1.03 GiB
2032	England/west-sussex	100	419,347	931,573	350.32 MiB	9 sec	3 sec	1.03 GiB
2039	England/west-sussex	100	442,292	958,567	356.98 MiB	9 sec	3 sec	1.04 GiB
2012	England/west-yorkshire	299	921,242	2,271,833	893.92 MiB	29 sec	15 sec	3.05 GiB
2020	England/west-yorkshire	299	963,460	2,339,931	930.52 MiB	29 sec	16 sec	3.12 GiB
2022	England/west-yorkshire	299	1,021,830	2,434,902	945.81 MiB	30 sec	16 sec	3.13 GiB
2032	England/west-yorkshire	299	1,021,830	2,434,902	945.81 MiB	30 sec	16 sec	3.13 GiB
2039	England/west-yorkshire	299	1,053,859	2,481,358	957.44 MiB	31 sec	16 sec	3.32 GiB
2012	England/wiltshire	89	285,600	704,491	274.63 MiB	7 sec	2 sec	921.03 MiB
2020	England/wiltshire	89	309,159	735,088	288.25 MiB	8 sec	2 sec	947.38 MiB
2022	England/wiltshire	89	335,400	774,105	292.74 MiB	8 sec	2 sec	949.12 MiB
2032	England/wiltshire	89	335,400	774,105	292.74 MiB	8 sec	2 sec	949.12 MiB
2039	England/wiltshire	89	348,866	792,075	296.45 MiB	8 sec	2 sec	955.03 MiB
2012	England/worcestershire	85	240,958	578,628	221.50 MiB	6 sec	2 sec	770.52 MiB
2020	England/worcestershire	85	255,594	601,116	231.62 MiB	7 sec	2 sec	790.33 MiB
2022	England/worcestershire	85	274,309	644,922	242.01 MiB	7 sec	2 sec	849.75 MiB
2032	England/worcestershire	85	274,309	644,922	242.01 MiB	7 sec	2 sec	849.75 MiB
2039	England/worcestershire	85	283,275	666,303	248.40 MiB	7 sec	2 sec	861.28 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Runtime	(Commute)	Memory_use
2012	Scotland/argyll-and-west-dunbartonshire	41	82,845	176,560	74.08 MiB	11 sec	1 sec	238.90 MiB
2020	Scotland/argyll-and-west-dunbartonshire	41	85,066	174,197	73.18 MiB	11 sec	1 sec	236.56 MiB
2022	Scotland/argyll-and-west-dunbartonshire	41	85,263	172,737	72.59 MiB	11 sec	1 sec	235.57 MiB
2032	Scotland/argyll-and-west-dunbartonshire	41	85,398	165,068	67.76 MiB	11 sec	1 sec	224.69 MiB
2039	Scotland/argyll-and-west-dunbartonshire	41	84,758	159,196	65.25 MiB	11 sec	1 sec	219.77 MiB
2012	Scotland/ayrshire	93	168,387	370,588	146.33 MiB	9 sec	1 sec	483.77 MiB
2020	Scotland/ayrshire	93	133,922	283,894	112.46 MiB	8 sec	1 sec	416.08 MiB
2022	Scotland/ayrshire	93	173,199	367,016	143.70 MiB	9 sec	1 sec	476.04 MiB
2032	Scotland/ayrshire	93	174,290	356,750	137.29 MiB	9 sec	1 sec	462.30 MiB
2039	Scotland/ayrshire	93	173,349	347,174	133.28 MiB	9 sec	1 sec	455.01 MiB
2012	Scotland/dumfries-and-galloway	40	68,416	149,648	61.42 MiB	6 sec	1 sec	217.04 MiB
2020	Scotland/dumfries-and-galloway	40	70,212	148,123	60.21 MiB	6 sec	1 sec	213.17 MiB
2022	Scotland/dumfries-and-galloway	40	70,455	147,351	59.47 MiB	6 sec	1 sec	211.49 MiB
2032	Scotland/dumfries-and-galloway	40	70,840	142,418	56.10 MiB	6 sec	1 sec	204.07 MiB
2039	Scotland/dumfries-and-galloway	40	70,668	138,573	54.77 MiB	6 sec	1 sec	202.05 MiB
2012	Scotland/edinburgh	111	225,093	497,378	186.98 MiB	7 sec	2 sec	555.70 MiB
2020	Scotland/edinburgh	111	242,994	525,476	198.41 MiB	8 sec	2 sec	732.84 MiB
2022	Scotland/edinburgh	111	248,491	532,384	200.96 MiB	8 sec	2 sec	738.35 MiB
2032	Scotland/edinburgh	111	273,234	562,902	207.62 MiB	8 sec	2 sec	791.61 MiB
2039	Scotland/edinburgh	111	288,360	578,847	210.49 MiB	8 sec	2 sec	793.17 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Pa	ntime(C	ommute)	Memory_use
2012	Scotland/fife	104	162,121	368,038	145.78 MiB	6 sec	1 sec		484.35 MiB
2020	Scotland/fife	104	159,563	371,896	147.05 MiB	6 sec	1 sec		486.65 MiB
2022	Scotland/fife	104	159,580	371,743	146.38 MiB	6 sec	1 sec		485.15 MiB
2032	Scotland/fife	104	166,255	370,447	141.66 MiB	6 sec	1 sec		472.29 MiB
2039	Scotland/fife	104	169,335	366,438	138.24 MiB	6 sec	1 sec		463.01 MiB
2012	Scotland/forth-valley	78	130,141	302,504	121.15 MiB	8 sec	1 sec		414.67 MiB
2020	Scotland/forth-valley	78	136,735	308,153	122.32 MiB	8 sec	1 sec		436.38 MiB
2022	Scotland/forth-valley	78	138,447	310,297	122.89 MiB	8 sec	1 sec		437.80 MiB
2032	Scotland/forth-valley	78	146,138	318,438	122.93 MiB	8 sec	1 sec		435.84 MiB
2039	Scotland/forth-valley	78	150,069	322,395	123.80 MiB	8 sec	1 sec		436.43 MiB
2012	Scotland/greater-glasgow	184	368,013	805,502	306.63 MiB	11 sec	4 sec		985.47 MiB
2020	Scotland/greater-glasgow	184	382,846	836,875	320.55 MiB	11 sec	4 sec		1013.11 MiB
2022	Scotland/greater-glasgow	184	388,050	842,636	322.55 MiB	11 sec	4 sec		1017.20 MiB
2032	Scotland/greater-glasgow	184	411,534	866,464	327.49 MiB	11 sec	4 sec		1.00 GiB
2039	Scotland/greater-glasgow	184	427,529	880,981	329.51 MiB	11 sec	4 sec		1023.96 MiB
2012	Scotland/highlands-and-islands	78	136,249	305,988	140.72 MiB	56 sec	1 sec		451.01 MiB
2020	Scotland/highlands-and-islands	78	144,639	307,886	140.39 MiB	57 sec	1 sec		447.70 MiB
2022	Scotland/highlands-and-islands	78	145,837	307,923	139.70 MiB	57 sec	1 sec		445.96 MiB
2032	Scotland/highlands-and-islands	78	149,761	305,422	135.12 MiB	56 sec	1 sec		434.37 MiB
2039	Scotland/highlands-and-islands	78	150,652	301,591	133.25 MiB	56 sec	1 sec		430.68 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Pb_runtime	(C_commute)	Memory_use
2012	Scotland/lanarkshire	160	287,147	654,563	258.58 MiB	11 sec	2 sec	903.22 MiB
2020	Scotland/lanarkshire	160	302,111	661,042	261.24 MiB	11 sec	2 sec	906.74 MiB
2022	Scotland/lanarkshire	160	305,554	662,692	261.37 MiB	11 sec	2 sec	907.35 MiB
2032	Scotland/lanarkshire	160	318,581	667,589	257.31 MiB	11 sec	2 sec	895.50 MiB
2039	Scotland/lanarkshire	160	324,614	666,795	254.59 MiB	11 sec	2 sec	887.40 MiB
2012	Scotland/north-east	132	250,789	587,273	228.59 MiB	14 sec	2 sec	795.80 MiB
2020	Scotland/north-east	132	267,964	586,245	230.01 MiB	14 sec	2 sec	841.08 MiB
2022	Scotland/north-east	132	271,745	587,957	230.81 MiB	14 sec	2 sec	842.86 MiB
2032	Scotland/north-east	132	287,988	594,876	228.56 MiB	14 sec	2 sec	836.51 MiB
2039	Scotland/north-east	132	297,440	594,445	226.47 MiB	14 sec	2 sec	830.82 MiB
2012	Scotland/renfrewshire-and-inverclyde	55	119,057	254,125	99.98 MiB	5 sec	1 sec	293.66 MiB
2020	Scotland/renfrewshire-and-inverclyde	55	124,460	256,040	100.44 MiB	5 sec	1 sec	293.33 MiB
2022	Scotland/renfrewshire-and-inverclyde	55	125,450	256,087	100.34 MiB	5 sec	1 sec	293.55 MiB
2032	Scotland/renfrewshire-and-inverclyde	55	129,185	255,008	97.93 MiB	5 sec	1 sec	287.17 MiB
2039	Scotland/renfrewshire-and-inverclyde	55	131,507	252,677	96.59 MiB	5 sec	1 sec	306.43 MiB
2012	Scotland/tayside	92	186,890	414,921	162.38 MiB	10 sec	1 sec	513.43 MiB
2020	Scotland/tayside	92	195,140	416,793	162.39 MiB	10 sec	1 sec	510.25 MiB
2022	Scotland/tayside	92	197,192	416,846	162.22 MiB	10 sec	1 sec	510.05 MiB
2032	Scotland/tayside	92	205,693	415,175	158.45 MiB	10 sec	1 sec	501.29 MiB
2039	Scotland/tayside	92	210,290	411,445	156.35 MiB	10 sec	1 sec	497.39 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Routing	(Commute)	Memory_use
2012	Scotland/the-lothians-and-scottish-borders	111	205,879	482,896	194.90 MiB	12 sec	2 sec	580.86 MiB
2020	Scotland/the-lothians-and-scottish-borders	111	223,446	501,223	201.50 MiB	12 sec	2 sec	590.52 MiB
2022	Scotland/the-lothians-and-scottish-borders	111	227,783	507,880	203.76 MiB	12 sec	2 sec	595.27 MiB
2032	Scotland/the-lothians-and-scottish-borders	111	246,603	537,145	210.28 MiB	12 sec	2 sec	761.01 MiB
2039	Scotland/the-lothians-and-scottish-borders	111	257,299	552,545	214.47 MiB	12 sec	2 sec	767.17 MiB
2012	Wales/bridgend-and-neath-port-talbot	38	119,725	283,159	108.22 MiB	4 sec	1 sec	382.14 MiB
2020	Wales/bridgend-and-neath-port-talbot	38	123,909	289,896	111.11 MiB	4 sec	1 sec	387.34 MiB
2022	Wales/bridgend-and-neath-port-talbot	38	124,921	292,227	111.51 MiB	4 sec	1 sec	387.62 MiB
2032	Wales/bridgend-and-neath-port-talbot	38	128,601	301,529	113.58 MiB	4 sec	1 sec	390.72 MiB
2039	Wales/bridgend-and-neath-port-talbot	38	129,740	307,260	114.33 MiB	4 sec	1 sec	391.18 MiB
2012	Wales/cardiff-and-vale-of-glamorgan	63	199,208	484,182	187.22 MiB	5 sec	1 sec	558.11 MiB
2020	Wales/cardiff-and-vale-of-glamorgan	63	214,676	499,272	194.75 MiB	5 sec	1 sec	572.81 MiB
2022	Wales/cardiff-and-vale-of-glamorgan	63	218,981	502,763	196.15 MiB	5 sec	1 sec	575.96 MiB
2032	Wales/cardiff-and-vale-of-glamorgan	63	240,112	522,526	199.47 MiB	5 sec	1 sec	577.76 MiB
2039	Wales/cardiff-and-vale-of-glamorgan	63	254,162	531,549	201.86 MiB	6 sec	1 sec	737.22 MiB
2012	Wales/central-valleys	38	124,691	296,581	115.15 MiB	4 sec	1 sec	396.09 MiB
2020	Wales/central-valleys	38	130,072	301,907	117.77 MiB	4 sec	1 sec	400.86 MiB
2022	Wales/central-valleys	38	131,383	303,557	118.40 MiB	4 sec	1 sec	424.36 MiB
2032	Wales/central-valleys	38	136,404	310,032	118.04 MiB	4 sec	1 sec	421.02 MiB
2039	Wales/central-valleys	38	138,735	314,703	119.17 MiB	4 sec	1 sec	422.91 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Pb_runtime	(Commute)	Memory_use
2012	Wales/conwy-and-denbighshire	30	92,732	211,205	80.51 MiB	4 sec	1 sec	251.37 MiB
2020	Wales/conwy-and-denbighshire	30	95,314	213,302	81.57 MiB	4 sec	1 sec	253.52 MiB
2022	Wales/conwy-and-denbighshire	30	95,881	214,182	81.86 MiB	4 sec	1 sec	254.11 MiB
2032	Wales/conwy-and-denbighshire	30	97,683	218,122	81.12 MiB	4 sec	1 sec	251.06 MiB
2039	Wales/conwy-and-denbighshire	30	97,687	220,933	80.93 MiB	4 sec	1 sec	249.66 MiB
2012	Wales/flintshire-and-wrexham	38	122,180	288,696	113.33 MiB	4 sec	1 sec	393.53 MiB
2020	Wales/flintshire-and-wrexham	38	127,660	292,056	114.59 MiB	4 sec	1 sec	395.17 MiB
2022	Wales/flintshire-and-wrexham	38	129,007	292,644	115.04 MiB	4 sec	1 sec	396.45 MiB
2032	Wales/flintshire-and-wrexham	38	134,527	292,817	112.38 MiB	4 sec	1 sec	410.81 MiB
2039	Wales/flintshire-and-wrexham	38	136,425	293,540	112.23 MiB	4 sec	1 sec	410.67 MiB
2012	Wales/gwent-valleys	46	144,178	341,543	132.18 MiB	4 sec	1 sec	450.92 MiB
2020	Wales/gwent-valleys	46	148,386	344,566	132.84 MiB	4 sec	1 sec	450.78 MiB
2022	Wales/gwent-valleys	46	149,374	345,498	132.73 MiB	4 sec	1 sec	450.12 MiB
2032	Wales/gwent-valleys	46	151,842	347,976	130.51 MiB	4 sec	1 sec	442.75 MiB
2039	Wales/gwent-valleys	46	151,729	350,397	130.60 MiB	4 sec	1 sec	442.92 MiB
2012	Wales/gwynedd	17	52,926	122,595	48.30 MiB	3 sec	1 sec	141.40 MiB
2020	Wales/gwynedd	17	55,064	124,569	49.30 MiB	3 sec	1 sec	143.64 MiB
2022	Wales/gwynedd	17	55,683	125,030	49.22 MiB	3 sec	1 sec	143.38 MiB
2032	Wales/gwynedd	17	58,372	128,844	49.83 MiB	3 sec	1 sec	143.73 MiB
2039	Wales/gwynedd	17	59,746	130,948	50.66 MiB	3 sec	1 sec	145.55 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Pa_runtime	(C_runtime)	Memory_use
2012	Wales/isle-of-anglesey	9	30,797	69,919	27.65 MiB	3 sec	1 sec	96.69 MiB
2020	Wales/isle-of-anglesey	9	31,366	69,845	27.85 MiB	3 sec	1 sec	97.28 MiB
2022	Wales/isle-of-anglesey	9	31,488	69,864	27.91 MiB	3 sec	1 sec	97.60 MiB
2032	Wales/isle-of-anglesey	9	31,601	69,502	27.10 MiB	3 sec	1 sec	95.40 MiB
2039	Wales/isle-of-anglesey	9	31,337	69,423	26.91 MiB	3 sec	1 sec	95.26 MiB
2012	Wales/monmouthshire-and-newport	31	100,402	240,491	94.45 MiB	4 sec	1 sec	280.30 MiB
2020	Wales/monmouthshire-and-newport	31	104,394	250,185	98.12 MiB	4 sec	1 sec	286.88 MiB
2022	Wales/monmouthshire-and-newport	31	105,481	253,282	99.28 MiB	4 sec	1 sec	288.93 MiB
2032	Wales/monmouthshire-and-newport	31	109,752	265,785	102.22 MiB	4 sec	1 sec	371.30 MiB
2039	Wales/monmouthshire-and-newport	31	111,246	273,319	103.91 MiB	4 sec	1 sec	373.72 MiB
2012	Wales/powys	19	59,028	132,725	51.23 MiB	4 sec	1 sec	184.96 MiB
2020	Wales/powys	19	59,972	132,328	50.62 MiB	4 sec	1 sec	183.27 MiB
2022	Wales/powys	19	60,190	132,467	50.48 MiB	4 sec	1 sec	182.78 MiB
2032	Wales/powys	19	59,586	133,010	49.65 MiB	4 sec	1 sec	180.54 MiB
2039	Wales/powys	19	57,969	133,514	49.37 MiB	4 sec	1 sec	179.70 MiB
2012	Wales/south-west-wales	50	165,004	383,260	145.80 MiB	5 sec	1 sec	474.24 MiB
2020	Wales/south-west-wales	50	170,327	385,937	146.54 MiB	5 sec	1 sec	474.39 MiB
2022	Wales/south-west-wales	50	171,623	386,901	147.01 MiB	5 sec	1 sec	476.02 MiB
2032	Wales/south-west-wales	50	175,897	392,107	145.21 MiB	5 sec	1 sec	469.23 MiB
2039	Wales/south-west-wales	50	176,482	394,303	144.54 MiB	5 sec	1 sec	467.40 MiB

Year	Area	MSOAs	Households	Individuals	Pb_file_size	Pa_runtime	(C_commute)	Memory_use
2012	Wales/swansea	31	104,423	242,128	93.14 MiB	4 sec	1 sec	276.08 MiB
2020	Wales/swansea	31	110,304	247,820	95.76 MiB	4 sec	1 sec	281.31 MiB
2022	Wales/swansea	31	111,940	249,098	96.15 MiB	4 sec	1 sec	282.09 MiB
2032	Wales/swansea	31	119,141	257,653	98.32 MiB	4 sec	1 sec	285.46 MiB
2039	Wales/swansea	31	123,450	262,306	99.97 MiB	4 sec	1 sec	366.54 MiB
2012	special/birmingham	132	410,243	1,104,216	450.75 MiB	14 sec	5 sec	1.55 GiB
2020	special/birmingham	132	429,124	1,148,426	470.60 MiB	14 sec	5 sec	1.59 GiB
2022	special/birmingham	132	434,527	1,156,702	473.72 MiB	15 sec	5 sec	1.59 GiB
2032	special/birmingham	132	467,993	1,198,716	479.63 MiB	15 sec	5 sec	1.59 GiB
2039	special/birmingham	132	492,029	1,230,211	489.58 MiB	16 sec	5 sec	1.61 GiB
2012	special/liverpool	61	207,217	479,774	182.06 MiB	7 sec	1 sec	538.83 MiB
2020	special/liverpool	61	224,431	503,264	193.74 MiB	7 sec	1 sec	562.01 MiB
2022	special/liverpool	61	241,366	536,264	206.67 MiB	7 sec	1 sec	742.97 MiB
2032	special/liverpool	61	241,366	536,264	206.67 MiB	7 sec	1 sec	742.97 MiB
2039	special/liverpool	61	251,435	549,857	211.22 MiB	7 sec	1 sec	751.45 MiB
2012	special/manchester	57	204,775	525,548	207.38 MiB	10 sec	2 sec	752.26 MiB
2020	special/manchester	57	220,664	551,613	221.09 MiB	10 sec	2 sec	780.27 MiB
2022	special/manchester	57	241,262	576,313	226.35 MiB	10 sec	2 sec	785.85 MiB
2032	special/manchester	57	241,262	576,313	226.35 MiB	10 sec	2 sec	785.84 MiB
2039	special/manchester	57	253,464	589,904	230.46 MiB	11 sec	2 sec	793.05 MiB

Year	Area	MSOAs	Households	Individuals	pb_file_size	Runtime	(Commuting)	Memory_use
2012	special/northwest_transpennine	829	2,653,096	6,416,497	2.45 GiB	3 min	2 min	7.74 GiB
2020	special/northwest_transpennine	829	2,788,624	6,616,117	2.56 GiB	3 min	2 min	7.95 GiB
2022	special/northwest_transpennine	829	2,960,285	6,908,374	2.62 GiB	3 min	2 min	8.02 GiB
2032	special/northwest_transpennine	829	2,960,285	6,908,374	2.62 GiB	3 min	2 min	8.02 GiB
2039	special/northwest_transpennine	829	3,058,114	7,059,122	2.66 GiB	3 min	2 min	8.09 GiB
2012	special/oxford	18	55,081	154,065	61.14 MiB	4 sec	1 sec	207.79 MiB
2020	special/oxford	18	55,235	153,045	61.53 MiB	4 sec	1 sec	208.41 MiB
2022	special/oxford	18	56,840	149,534	58.11 MiB	4 sec	1 sec	199.69 MiB
2032	special/oxford	18	56,840	149,534	58.11 MiB	4 sec	1 sec	199.69 MiB
2039	special/oxford	18	58,038	147,239	56.67 MiB	4 sec	1 sec	196.62 MiB
2012	special/oxford_cambridge_arc	353	1,112,235	2,828,466	1.08 GiB	40 sec	21 sec	3.61 GiB
2020	special/oxford_cambridge_arc	353	1,199,021	2,950,743	1.14 GiB	41 sec	21 sec	3.73 GiB
2022	special/oxford_cambridge_arc	353	1,296,471	3,107,289	1.17 GiB	43 sec	22 sec	3.77 GiB
2032	special/oxford_cambridge_arc	353	1,314,402	3,122,071	1.17 GiB	43 sec	22 sec	3.76 GiB
2039	special/oxford_cambridge_arc	353	1,372,547	3,189,664	1.18 GiB	44 sec	23 sec	3.78 GiB

Notes:

- `pb_file_size` refers to the size of the uncompressed protobuf file in `data/output/`
- The total `runtime` is usually dominated by matching workers to businesses, so `commuting_runtime` gives a breakdown
- Measuring memory usage of Linux processes isn't straightforward, so `memory_usage` should just be a guide
- These measurements were all taken on one developer's laptop, and they don't represent multiple runs. This table just aims to give a general sense of how long running takes.

  - That machine has 10 cores, which matters for the parallelized commuting calculation.

- The time *usually* doesn't include downloading or decompressing raw data. For some areas, it might!
- `scripts/collect_stats.py` produces the table above