Lecture 5

Introduction to spatial modelling – spatial data types

Janine Illian
University of Glasgow
janine.illian@glasgow.ac.uk

October 30, 2025

Outline

- Why spatial modelling?
- Why is spatial modelling computationally expensive?
- Different data types:
  - Modelling in discrete space – areal data
  - Modelling in continuous space – geo-referenced data
  - Modelling continuous space – spatial point process data
- representation of data structures in `R`

Spatial modelling

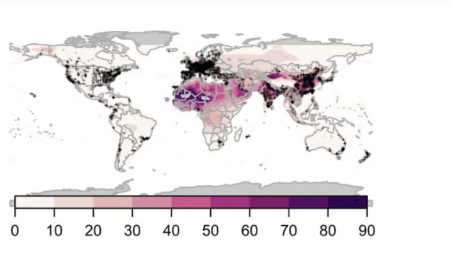
Many natural processes take place in space. Large amounts of data collected in space; increased resolution $\rightarrow$ large, complex datasets.

Challenges:

- Inaccessible to practitioners (literature aimed at statisticians)
- Methodology not always linked to applications
- Models may be too simple to reflect real-life data
- Difficult to apply without expertise

We will see that `inlabru` can help.

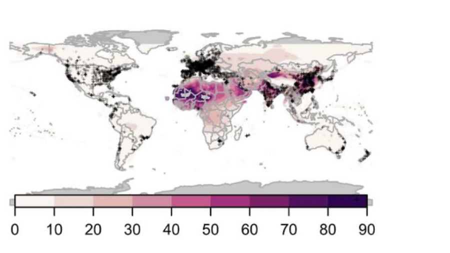
Example: Global PM 2.5



Exposure to air pollution; particulate matter $< 2.5\mu m$ (PM 2.5)

- Linked to poor health outcomes; responsible for 3 million deaths worldwide each year
- Observations potentially not independent
- Sparsely measured
- Heterogeneous spatial coverage

Example: Global PM 2.5



Exposure to air pollution; particulate matter $< 2.5\mu m$ (PM 2.5)

- Linked to poor health outcomes; responsible for 3 million deaths worldwide each year
- Observations potentially not independent
- Sparsely measured
- Heterogeneous spatial coverage

spatial dependence + complex observation processes

Spatial modelling: why necessary?

- Standard models assume independent observations
- Spatio-temporal data are often not independent
- Two nearby observations are similar $\rightarrow$ not providing independent info

Consequences of ignoring spatial dependence

Ignoring spatial dependence:

- pretending we have as much independent information as observations
- pretending we have more information than actually available

Consequences of ignoring spatial dependence

Ignoring spatial dependence:

- pretending we have as much independent information as observations
- pretending we have more information than actually available

$\rightsquigarrow$ spurious tight confidence intervals

$\rightsquigarrow$ wrong inference and conclusions

Consequences of ignoring spatial dependence

Ignoring spatial dependence:

- pretending we have as much independent information as observations
- pretending we have more information than actually available

$\rightsquigarrow$ spurious tight confidence intervals

$\rightsquigarrow$ wrong inference and conclusions

Spatial models include special components to explicitly model dependence.

Spatial modelling: computations

- computationally expensive
- in the past: often MCMC $\rightarrow$ very slow

Spatial modelling: aims

aims vary among different studies:

- describing spatial patterns and structures
- understanding spatial patterns and structures
- linking data observed in space to covariates while accounting for spatial autocorrelation
- predicting into unobserved locations

Spatial modelling: aims

aims vary among different studies:

- describing spatial patterns and structures
- understanding spatial patterns and structures
- linking data observed in space to covariates while accounting for spatial autocorrelation
- predicting into unobserved locations

aims also vary with different types of spatial data...

Types of spatial data

Jafet's fancy slide with the three data types

Types of spatial data

Discrete space:

- Data on a spatial grid (areal data)

Continuous space:

- Geostatistical (geo-referenced) data
- Spatial point data

Model components are used to reflect spatial dependence structures in discrete and continuous space.

Discrete space: areal data

- Data on a (regular or irregular) spatial grid
- Examples: number of individuals in a region, average rainfall in a province
- Originally point or geostatistical data; gridded for practical reasons

Observed response(s): Measurement associated with each grid cell or areal unit

Discrete space: areal data

- Data on a (regular or irregular) spatial grid
- Examples: number of individuals in a region, average rainfall in a province
- Originally point or geostatistical data; gridded for practical reasons

Observed response(s): Measurement associated with each grid cell or areal unit
here: spatial structure represented by Gauss Markov random field (GMRF)

the data we will use

respiratory hospitalisations in Glasgow:

- areal data on respiratory hospitalisations
- 271 Intermediate Zones (IZ) in Greater Glasgow and Clyde health board
- covariates concern air pollution concentration and socio-economic deprivation cotland

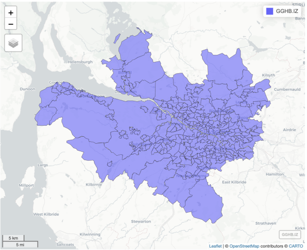


Figure 1: Greater Glasgow and Clyde health board represented by 271 Intermediate Zones

Standardized Mortality Ratios (SMR)

Response here:
disease risk, estimated using **Standardized Mortality Ratios (SMR)**: for spatial areal unit i SMR is the ratio between the observed (Y_i) and expected (E_i) number of cases:

$$SMR_i = \frac{Y_i}{E_i}$$

Standardized Mortality Ratios (SMR)

Response here:
disease risk, estimated using **Standardized Mortality Ratios (SMR)**: for spatial areal unit i SMR is the ratio between the observed (Y_i) and expected (E_i) number of cases:

$$SMR_i = \frac{Y_i}{E_i}$$

$SMR > 1$: more observed cases than expected
$\rightsquigarrow$ **high risk area** $SMR < 1$, fewer observed cases than expected
$\rightsquigarrow$ **low risk area**

Continuous space: geostatistical data

- phenomenon continuous in space
- measured at a finite set of locations
- examples: nutrient levels in soil, salinity in the sea

Observed response(s): measurements at given locations in continuous space

Continuous space: geostatistical data

- phenomenon continuous in space
- measured at a finite set of locations
- examples: nutrient levels in soil, salinity in the sea

Observed response(s): measurements at given locations in continuous space
here: spatial structure represented by Gaussian random field;

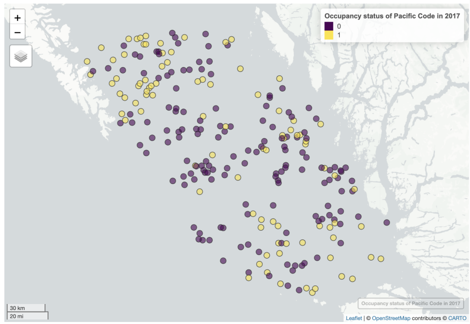
Continuous space: geostatistical data

- phenomenon continuous in space
- measured at a finite set of locations
- examples: nutrient levels in soil, salinity in the sea

Observed response(s): measurements at given locations in continuous space
here: spatial structure represented by Gaussian random field; approximated by a continuously indexed Gauss Markov random field

the data we will use

- Pacific Cod (Gadus macrocephalus)
    - trawl survey in Queen Charlotte Sound



detail on pacific cod data

Continuous space: spatial point patterns

- Locations of objects/events in space (typically 2D)
- Examples: tree locations, animal groups, earthquakes

Observed response(s): x,y coordinates (sometimes also additional measurements,; "marks")
modelled by a random variable, a spatial point process characterised by intensity $\lambda(s) s \in R$

Continuous space: spatial point patterns

- Locations of objects/events in space (typically 2D)
- Examples: tree locations, animal groups, earthquakes

Observed response(s): x,y coordinates (sometimes also additional measurements,; "marks")
modelled by a random variable, a spatial point process characterised by intensity $\lambda(s) s \in R$
here: intensity is a Gaussian random field;

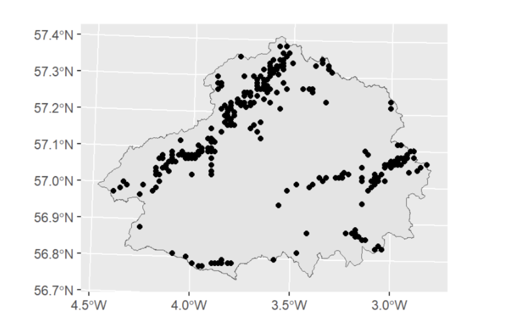
Continuous space: spatial point patterns

- Locations of objects/events in space (typically 2D)
- Examples: tree locations, animal groups, earthquakes

Observed response(s): x,y coordinates (sometimes also additional measurements,; "marks")
modelled by a random variable, a spatial point process
characterised by intensity $\lambda(s) s \in R$
here: intensity is a Gaussian random field;
approximated by a continuously indexed Gauss Markov random field

the data we will use

- citizen science data; species distribution modelling
- ringlet butterfly Scotland's Cairngorms National Park



Different spatial data structures...

- different types of spatial data structure – **modelling:**
    - modelled with different statistical models

Different spatial data structures...

- different types of spatial data structure – **modelling:**
  - modelled with different statistical models
  - used to answer different questions
  - should not be mixed up
- different types of spatial data structure – **computation:**
  - spatial structures can all be approximated by GMRF – the SPDE approach

Different spatial data structures...

- different types of spatial data structure – **modelling:**
  - modelled with different statistical models
  - used to answer different questions
  - should not be mixed up
- different types of spatial data structure – **computation:**
  - spatial structures can all be approximated by GMRF – the SPDE approach
  - GMRF can be efficiently fitted with INLA/inlabru
  - models can be efficiently fitted with INLA/inlabru
  - can be modelled within the same framework
  - different data structures can be jointly modelled

Point patterns vs. geostatistical data

Point patterns:

- Data format: x,y coordinates
- Optional: marks
- Aim: model locations as random

Geostatistical data:

- Data format: x,y coordinates
- Measurements mandatory
- Aim: model continuous process at measured locations

take-home message

- all spatial models discussed are special cases of Latent Gaussian models
- different spatial terms are needed for different spatial data structures
- the SPDE approach provide unifies approximation as all data structures are spatially referenced in continuous space
- `inlabru` efficiently fits these models

What happens next?

Practical 4: introduction to spatial data wrangling and manipulation

- exploring tools for visualization and wrangling
- using metrics to assess spatial autocorrelation in different types of data structure

Spatial data structures - the `sf` package

"Simple Features" for R (`sf`): modern, efficient framework for working with spatial data

- implements the **Simple Features – sf** standard for spatial vector data

Spatial data structures - the `sf` package

"Simple Features" for R (`sf`): modern, efficient framework for working with spatial data

- implements the **Simple Features – sf** standard for spatial vector data
- represents geometries (`POINT`, `LINESTRING`, `POLYGON`, etc.) as list-columns in data frames

Spatial data structures - the `sf` package

"Simple Features" for R (`sf`): modern, efficient framework for working with spatial data

- implements the **Simple Features** – **sf** standard for spatial vector data
- represents geometries (`POINT`, `LINESTRING`, `POLYGON`, etc.) as list-columns in data frames
- stores both **attributes** and **geometry** in a single object – a spatially-enabled data frame

Spatial data structures - the `sf` package

"Simple Features" for R (`sf`): modern, efficient framework for working with spatial data

- implements the **Simple Features – sf** standard for spatial vector data
- represents geometries (`POINT`, `LINESTRING`, `POLYGON`, etc.) as list-columns in data frames
- stores both **attributes** and **geometry** in a single object – a spatially-enabled data frame
- integrates seamlessly with `tidyverse` (e.g., `dplyr`, `ggplot2`)

Spatial data structures - the `sf` package

"Simple Features" for R (`sf`): modern, efficient framework for working with spatial data

- implements the **Simple Features – sf** standard for spatial vector data
- represents geometries (`POINT`, `LINESTRING`, `POLYGON`, etc.) as list-columns in data frames
- stores both **attributes** and **geometry** in a single object – a spatially-enabled data frame
- integrates seamlessly with `tidyverse` (e.g., `dplyr`, `ggplot2`)
- uses fast, modern spatial operations

Spatial data structures - the `sf` package

"Simple Features" for R (`sf`): modern, efficient framework for working with spatial data

- implements the **Simple Features – sf** standard for spatial vector data
- represents geometries (`POINT`, `LINESTRING`, `POLYGON`, etc.) as list-columns in data frames
- stores both **attributes** and **geometry** in a single object – a spatially-enabled data frame
- integrates seamlessly with `tidyverse` (e.g., `dplyr`, `ggplot2`)
- uses fast, modern spatial operations
- **not** specific to INLA/inlabru

Spatial data structures - the `sf` package

"Simple Features" for R (`sf`): modern, efficient framework for working with spatial data

- implements the **Simple Features – sf** standard for spatial vector data
- represents geometries (`POINT`, `LINESTRING`, `POLYGON`, etc.) as list-columns in data frames
- stores both **attributes** and **geometry** in a single object – a spatially-enabled data frame
- integrates seamlessly with `tidyverse` (e.g., `dplyr`, `ggplot2`)
- uses fast, modern spatial operations
- **not** specific to INLA/inlabru
- replaces `sp`

`terra` package in R

`terra`: designed for working with spatial **raster** and **vector** data

Key Features

- handles large raster datasets efficiently
- provides a unified framework for raster and vector data structures
- supports common GIS operations:
  - reprojection, resampling, cropping, masking
  - raster and map algebra
  - zonal statistics, extraction, distance calculations
- integrates with `sf` for combined raster–vector workflows
- supports numerous spatial data formats via the GDAL library.
- successor to `raster` package; faster, more memory-efficient tools

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.

`terra`

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).

`terra`

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.

`terra`

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.
- Integrates seamlessly with the `tidyverse` (`dplyr`, `ggplot2`).

`terra`

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.
- Integrates seamlessly with the `tidyverse` (`dplyr`, `ggplot2`).

`terra`

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.
- Integrates seamlessly with the `tidyverse` (`dplyr`, `ggplot2`).

`terra`

- Handles **raster data**: continuous gridded surfaces (e.g., elevation, temperature).

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.
- Integrates seamlessly with the `tidyverse` (`dplyr`, `ggplot2`).

`terra`

- Handles **raster data**: continuous gridded surfaces (e.g., elevation, temperature).
- Replaces the older `raster` package.

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.
- Integrates seamlessly with the `tidyverse` (`dplyr`, `ggplot2`).

`terra`

- Handles **raster data**: continuous gridded surfaces (e.g., elevation, temperature).
- Replaces the older `raster` package.
- Efficient memory management and parallel processing.

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.
- Integrates seamlessly with the `tidyverse` (`dplyr`, `ggplot2`).

`terra`

- Handles **raster data**: continuous gridded surfaces (e.g., elevation, temperature).
- Replaces the older `raster` package.
- Efficient memory management and parallel processing.
- Can read, write, and process large spatial

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.
- Integrates seamlessly with the `tidyverse` (`dplyr`, `ggplot2`).

`terra`

- Handles **raster data**: continuous gridded surfaces (e.g., elevation, temperature).
- Replaces the older `raster` package.
- Efficient memory management and parallel processing.
- Can read, write, and process large spatial

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.
- Integrates seamlessly with the `tidyverse` (`dplyr`, `ggplot2`).

`terra`

- Handles **raster data**: continuous gridded surfaces (e.g., elevation, temperature).
- Replaces the older `raster` package.
- Efficient memory management and parallel processing.
- Can read, write, and process large spatial

Comparing `sf` and `terra` in R

Both `sf` and `terra` are modern R packages for working with spatial data, but they are designed for different types of spatial objects.

`sf` (*simple features*)

- Handles **vector data**: points, lines, polygons.
- Follows the Simple Features standard (ISO 19125).
- Stores geometry and attributes together in a data frame.
- Integrates seamlessly with the `tidyverse` (`dplyr`, `ggplot2`).

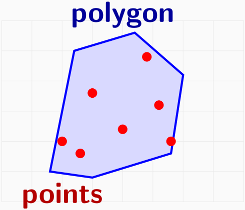
`terra`

- Handles **raster data**: continuous gridded surfaces (e.g., elevation, temperature).
- Replaces the older `raster` package.
- Efficient memory management and parallel processing.
- Can read, write, and process large spatial

vector vs. raster data:
`sf` vs. `terra`

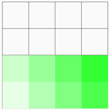
`sf`: vector data

- discrete geometric objects: points, lines, polygons
- store exact shapes and boundaries
- used for administrative areas, roads, sample sites



`terra`: raster data

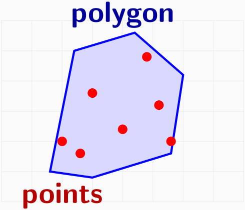
- continuous grid of cells (pixels)
- each cell stores a numeric value (e.g., elevation, temperature)
- used for spatially explicit covariates that have values averywhere in space



vector vs. raster data: `sf` vs. `terra`

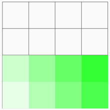
`sf`: vector data

- discrete geometric objects: points, lines, polygons
- store exact shapes and boundaries
- used for administrative areas, roads, sample sites



`terra`: raster data

- continuous grid of cells (pixels)
- each cell stores a numeric value (e.g., elevation, temperature)
- used for spatially explicit covariates that have values averywhere in space



further slides

`sf` **vs.** `sp`

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s

modern: `sf`

sf **vs.** sp

sf is a modern replacement of the sp package

old: sp

- introduced in early 2000s
- complex S4 class system

modern: sf

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes

modern: `sf`

`sf` vs. `sp`

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes
- limited integration with `dplyr`/`ggplot2`

modern: `sf`

`sf` vs. `sp`

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes
- limited integration with `dplyr`/`ggplot2`
- manual handling of coordinate reference systems

modern: `sf`

`sf` **vs.** `sp`

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes
- limited integration with `dplyr`/`ggplot2`
- manual handling of coordinate reference systems

modern: `sf`

- introduced in 2016, built on the Simple Features standard

`sf` vs. `sp`

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes
- limited integration with `dplyr`/`ggplot2`
- manual handling of coordinate reference systems

modern: `sf`

- introduced in 2016, built on the Simple Features standard
- uses tidyverse-friendly data frame structure

`sf` vs. `sp`

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes
- limited integration with `dplyr`/`ggplot2`
- manual handling of coordinate reference systems

modern: `sf`

- introduced in 2016, built on the Simple Features standard
- uses tidyverse-friendly data frame structure
- geometry stored as a list-column (`sfc`)

`sf` vs. `sp`

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes
- limited integration with `dplyr`/`ggplot2`
- manual handling of coordinate reference systems

modern: `sf`

- introduced in 2016, built on the Simple Features standard
- uses tidyverse-friendly data frame structure
- geometry stored as a list-column (`sfc`)
- works seamlessly with `dplyr`, `ggplot2`, `tidyr`

`sf` vs. `sp`

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes
- limited integration with `dplyr`/`ggplot2`
- manual handling of coordinate reference systems

modern: `sf`

- introduced in 2016, built on the Simple Features standard
- uses tidyverse-friendly data frame structure
- geometry stored as a list-column (`sfc`)
- works seamlessly with `dplyr`, `ggplot2`, `tidyr`
- automatic CRS handling and conversions

`sf` vs. `sp`

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes
- limited integration with `dplyr`/`ggplot2`
- manual handling of coordinate reference systems

modern: `sf`

- introduced in 2016, built on the Simple Features standard
- uses tidyverse-friendly data frame structure
- geometry stored as a list-column (`sfc`)
- works seamlessly with `dplyr`, `ggplot2`, `tidyr`
- automatic CRS handling and conversions

`sf` is a modern replacement of the `sp` package

old: `sp`

- introduced in early 2000s
- complex S4 class system
- separate objects for geometry and attributes
- limited integration with `dplyr`/`ggplot2`
- manual handling of coordinate reference systems

modern: `sf`

- introduced in 2016, built on the Simple Features standard
- uses tidyverse-friendly data frame structure
- geometry stored as a list-column (`sfc`)
- works seamlessly with `dplyr`, `ggplot2`, `tidyr`
- automatic CRS handling and conversions

`sf` simplifies workflows, speeds up computation, and integrates