sfislands: An R Package for Accommodating Islands and Disjoint Zones in Areal Spatial Modelling

by Kevin Horan, Katarina Domijan, and Chris Brunsdon

Abstract Fitting areal spatial models can be a cumbersome task, particularly when the geographical units are not well-behaved. The presence of islands gives rise to particular issues when creating neighbourhood structures based on contiguity. We might also want to edit the output of a condition-based neighbourhood structure. It might be appropriate to add or remove connections to take account of infrastructure links or other domain-specific issues which are not apparent from a basic set of polygons. This package streamlines these processes with tools to account for the presence of islands and to intuitively make manual alterations to neighbourhood structures with the help of convenient mapping functions. Beyond the setting up of a satisfactory neighbourhood structure, it has further helper functions specific to the `mgcv` package which reduce the human workload required to extract estimates from models with spatial and hierarchical structures.

1 Introduction

A key feature which differentiates spatial statistics is the non-independence of observations and the expectation that neighbouring units will be more similar than non-neighbouring ones (Tobler 1970). If this is not accounted for, the assumptions of many types of models will be violated. The relationships between all spatial units in a study can be represented numerically in a spatial weights matrix. In order to build this, we must first decide on what constitutes being a neighbour. We can see this as a continuous relationship where degree of neighbourliness is a function of connectivity, which could be represented as some measure of distance. Alternatively it could be a binary situation where each pair of units either are (1) or are not (0) neighbours. This can be based on a condition such as contiguity of some sort, or a distance constraint. It is the job of the modeller to formulate a hypothesis which justifies their choice of neighbourhood structure.

For R users, the **spdep** package (Bivand et al. 2005) has long been popular for the creation of these matrices. More recently, with the increasing use of **sf** structures and their compatibility with the **tidyverse** (Wickham et al. 2019), the **sfdep** (Pebesma 2018) package (Parry and Locke 2024) has presented the same functionality in addition to extra features in a tidy structure based on lists in a similar way to **sf**.

The most appropriate form of neighbourhood structure will depend on the specific context. Briz-Redón et al. (2021) compared different structures in the context of COVID-19 data. They note that Earnest et al. (2007) found that distance-based matrices were more appropriate when examining birth defects in Australia, whereas Duncan, White, and Mengersen (2017) found that a first-order contiguity structure produced a better fit than others in the context of lip cancer incidence in Scotland.

The most commonly used neighbourhood structure is one based on first-order queen contiguity, where units are considered neighbours if they share at least a vortex of boundary. However, as the name suggests, this will lead to problems when non-contiguous units such as islands or exclaves are present. Less obviously, depending on how the geographic units are described, areas on either sides of rivers may be inappropriately classified as neighbours or not neighbours. Furthermore, the presence of infrastructure such as tunnels, bridges or ferry services might be satisfactory to meet our hypothesis of the required degree of connectivity to be considered neighbours.

The aim of `sfislands` is to deal with the situations described above in a convenient and open manner. It allows us to set up a structure, quickly map it, and then examine it. The structure can then be edited and the process re-iterated until we have described a spatial relationship structure with which we are satisfied.

The above can be considered as the *pre-functions* of the package. A further set of feature of `sfislands`, which we refer to as *post-functions*, are for use with the **mgcv** package (Wood 2011). It is straightforward to use `sfislands` neighbourhood structures in models with **mgcv** but it can be quite awkward to extract the estimates. These *post-functions* conveniently provide the estimates in tidy format of random effects and intrinsic conditional autoregressive (ICAR) components from such models and give the ability to quickly map them for visualisation purposes.

Typical use-cases

In this paper, we will look at three examples to show different use-cases for `sfislands`. The first example focuses on earthquakes in Indonesia. It shows a scenario where all of the functions are used, from setting up contiguities to modelling and examining the estimates of the model.

The second example looks at London and how, despite the lack of islands, the presence of a river means that some of the pre-functions of `sfislands` can be useful.

The final example focuses on Liverpool. There are no islands or issues of discontiguity in this dataset. Instead, we will show how the package can be used to fit a multilevel model with an ICAR component at the lowest level. We show how the results can be presented in tidy form and quickly visualised. We also show how different types of contiguity from the **sfdep** package can used within the **mgcv** modelling framework and how the estimates derived from them can be conveniently compared.

2 Why use `sfislands`?

It greatly speeds up the workflow of fitting areal spatial models.

- **pre-functions** for setting up contiguities:
  - it addresses an issue commonly seen in online help forums where an inexperienced user wishes to get started with a model but fails at the first hurdle because their neighbourhood structure is not compatible with their data. `sfislands` will include a contiguity for all units,
  - it gives tools to immediately visualise this structure as a map,
  - these maps are created using **ggplot2** (Wickham 2016), which allows users to apply the desired styling and themes using **ggplot2** syntax,
  - as the nodes can be labelled by index, it makes it very easy to add and remove connections as appropriate with confidence,
  - connections made by the package which are not contiguities can be accessed to ensure openness in the process.
- For modelling:
  - these neighbourhood structures can be used in modelling packages such as **mgcv**, **brms**. (Bürkner 2017), **r-inla** (Bakka et al. 2018) and more.
- **post-functions** for `mgcv` models:
  - it simplifies the process of extracting estimates from models with random effects and Markov random field structures (ICAR models),
  - these effects can be quickly visualised in **ggplot2** maps.

3 Pre-functions

The first group of functions, shown in Table 1, deals with the creation of a contiguity structure in the presence of discontiguities. The resultant structure can be quickly mapped to check if it is satisfactory. Connections can be manually added or removed by name or index number. By an iterative process of changes and examination of a quick map, a satisfactory structure can be decided upon.

Table 1 shows these functions and their purpose.

They deal with the creation of a contiguity structure in the presence of discontiguities. The resultant structure can be quickly mapped to check if it is satisfactory. Connections can be manually added or removed by name or index number. By an iterative process of changes and examination of a quick map, a satisfactory structure can be decided upon. Going through each function in more detail:

Use the rectangles shown in Figure 1 as a simple demonstration. Rectangles 1-3 are contiguous while 4 and 5 are islands.

Table 1: Pre-functions: setting up a neighbourhood structure.

function	purpose
st_bridges()	create a neighbourhood contiguity structure, with a k-nearest neighbours condition for islands
st_quickmap_nb()	check structure visually on map
st_check_islands()	check the contiguities which have been assigned to islands
st_manual_join_nb()	make manual changes by adding connections
st_manual_cut_nb()	make manual changes by removing connections

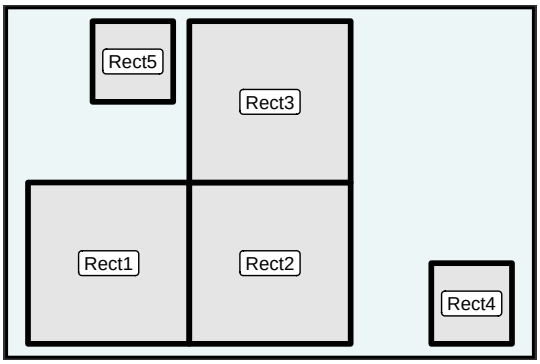

Figure 1: Simplified scenario with five rectangles.

st_bridges()

This function requires as arguments an **sf** dataframe and the name of a column of unique identifiers, preferably names, of each spatial unit. The k-nearest neighbours for connecting islands can then be specified. The output is a *named* neighbourhood structure in either list or matrix form as desired, which can be either a standalone object or included as an additional column in the original **sf** dataframe. While it is not necessary in all modelling packages for the neighbourhood list or matrix to be *named*, it is good practice to do so and is mandatory when using, for example, **mgcv**.

In the following examples, we have chosen to ignore islands with the argument `remove_islands = TRUE` and to return a list and matrix structure respectively by specifying this in the `nb_structure` argument and choosing `add_to_dataframe = FALSE`:

```
# output a named list

st_bridges(rectangles,
           "name",
           remove_islands = TRUE,
           nb_structure = "list",
           add_to_dataframe = FALSE) |>
  head()

#> $Rect1
#> [1] 2 3
#>
#> $Rect2
#> [1] 1 3
#>
#> $Rect3
#> [1] 1 2

# output a named matrix

st_bridges(rectangles,
           "name",
           remove_islands = TRUE,
           nb_structure = "matrix",
           add_to_dataframe = FALSE) |>
```

```
  head()

#>       [,1] [,2] [,3]
#> Rect1    0    1    1
#> Rect2    1    0    1
#> Rect3    1    1    0
```

In these examples, we choose to join islands to their 1 nearest neighbour, which is the default setting, and to return the output as a column called "nb" in the original sf dataframe (this is the default setting):

```
# output a named list as a column "nb" in original dataframe

st_bridges(rectangles,
           "name",
           link_islands_k = 1,
           nb_structure = "list") |>
  head()

#> Simple feature collection with 5 features and 2 fields
#> Geometry type: POLYGON
#> Dimension:     XY
#> Bounding box:  xmin: 0 ymin: 0 xmax: 6 ymax: 4
#> CRS:           NA
#>                          geometry  name      nb
#> 1 POLYGON ((0 0, 0 2, 2 2, 2 ... Rect1    2, 3
#> 2 POLYGON ((2 0, 2 2, 4 2, 4 ... Rect2 1, 3, 4
#> 3 POLYGON ((2 2, 2 4, 4 4, 4 ... Rect3 1, 2, 5
#> 4 POLYGON ((5 0, 5 1, 6 1, 6 ... Rect4       2
#> 5 POLYGON ((0.8 3, 0.8 4, 1.8... Rect5       3

# output a named matrix as a column "nb" in original dataframe

st_bridges(rectangles,
           "name",
           link_islands_k = 1,
           nb_structure = "matrix") |>
  head()

#> Simple feature collection with 5 features and 2 fields
#> Geometry type: POLYGON
#> Dimension:     XY
#> Bounding box:  xmin: 0 ymin: 0 xmax: 6 ymax: 4
#> CRS:           NA
#>                          geometry  name nb.1 nb.2 nb.3 nb.4 nb.5
#> 1 POLYGON ((0 0, 0 2, 2 2, 2 ... Rect1    0    1    1    0    0
#> 2 POLYGON ((2 0, 2 2, 4 2, 4 ... Rect2    1    0    1    1    0
#> 3 POLYGON ((2 2, 2 4, 4 4, 4 ... Rect3    1    1    0    0    1
#> 4 POLYGON ((5 0, 5 1, 6 1, 6 ... Rect4    0    1    0    0    0
#> 5 POLYGON ((0.8 3, 0.8 4, 1.8... Rect5    0    0    1    0    0
```

These structures can serve as the input to models in brms, **r-inla**, **rstan** (Stan Development Team 2020) or mgcv. Rather than having a separate neighbours column, it is included as a named list or matrix in the original sf dataframe, in the spirit of the sfdep package.

st_quickmap_nb()

It is much more intuitive to examine these structures visually than in matrix or list format. This can be done with the `st_quickmap_nb()` function as shown in Figure 2.

```
# default is 'nodes = "point"'

st_bridges(rectangles,
```

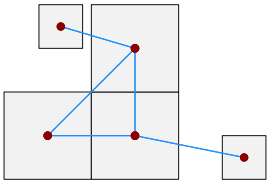

Figure 2: Queen contiguity and islands connected to nearest neighbour.

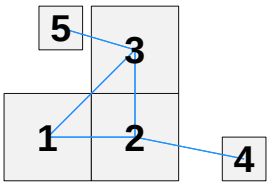

Figure 3: Queen contiguity and islands connected to nearest neighbour. Nodes are shown as numeric indices.

```
          "name",
          link_islands_k = 1) |>
  st_quickmap_nb()
```

If we wish to make edits, it might be more useful to represent the nodes numerically rather than as points (Figure 3).

```
# with 'nodes = "numeric"'

st_bridges(rectangles,
          "name",
          link_islands_k = 1) |>
  st_quickmap_nb(nodes = "numeric")
```

st_check_islands()

This function will tell us openly what connections have been made which are not based on contiguity. It gives both the name and index number of each pair of added connections.

```
st_bridges(rectangles,
          "name",
          link_islands_k = 1) |>
  st_check_islands()

#>   island_names island_num nb_num nb_names
#> 1        Rect4          4      2    Rect2
#> 2        Rect5          5      3    Rect3
```

st_manual_join_nb()

If we feel that 4 should also be connected to 3, this can be done manually (Figure 4).

```
st_bridges(rectangles, "name",
          link_islands_k = 1) |>
  st_manual_join_nb(3,4) |>
  st_quickmap_nb(nodes = "numeric")
```

st_manual_cut_nb()

And perhaps there is a wide river between rectangles 1 and 2 which justifies removing the connection. We will edit it this time using names (Figure 5).

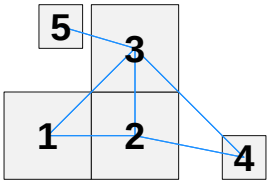

Figure 4: With an additional connection between 3 and 4.

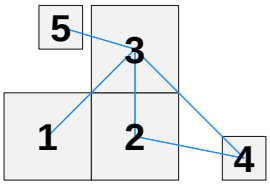

Figure 5: With an additional connection between 1 and 2.

```
st_bridges(rectangles, "name",
           link_islands_k = 1) |>
  st_manual_join_nb(3,4) |>
  st_manual_cut_nb("Rect1","Rect2") |>
  st_quickmap_nb(nodes = "numeric")
```

The use of these structures is particularly common in CAR-type models. These are often implemented in a Bayesian framework using **brms**, **r-inla** or **rstan**. The pre-functions will output the neighbourhood structure in the desired format for use in any of these frameworks. A convenient frequentist alternative is to use the **mgcv** package. It has the functionality to create such models using `bs="mrf"`. It also has the ability to combine these with a hierarchical structure using `bs="re"`. While the outputs from the Bayesian structures mentioned above can be extracted in the same way as any other component of the model, it can be a bit awkward to get the estimates from **mgcv** model. `sfislands` has two post-functions to conveniently extract and visualise these.

4 Post-functions

Table 2 shows the second set of functions in the package and their purpose.

st_augment()

This function augments the original dataframe with the estimated means and standard errors from the **mgcv** model in a similar manner to how the **broom** (Robinson, Hayes, and Couch 2023) package operates. The `geometry` column, as per convention, remains as the last column of the augmented dataframe, while the new random effects columns are positioned immediately before it. The estimates which `st_augment()` will extract are random effects (which are called in **mgcv** with `bs='re'`) and ICAR components (`bs='mrf'`).

Consider the model structure described in the code below using **mgcv** syntax. In this model y is the dependent variable while it is being estimated with a fixed intercept, a fixed slope for some covariate, a random intercept and slope for the covariate at a *region* level, and an ICAR varying intercept and slope at a lower *sub-region* level.

```
mgcv::gam(
  y ~ covariate +                  # fixed intercept and effect for covariate
    s(region, bs = "re") +                 # random intercept at level region
    s(region, covariate, bs = "re") +         # random slopes at level region
```

Table 2: Post-functions: tidy estimates from mgcv.

function	purpose
st_augment()	augment the original dataframe with model predictions
st_quickmap_preds()	generate quick maps of these predictions

Table 3: The naming procedure for augmented columns from different mgcv structures.

mgcv syntax	column name
s(region, bs = 're')	random.effect.region
s(region, covariate, bs = 're')	random.effect.covariate | region
s(sub-region, bs = 'mrf', xt = list(nb = data$nb)	mrf.smooth.sub-region
s(sub-region, by = covariate, bs = 'mrf', xt = list(nb = data$nb)	mrf.smooth.covariate | sub-region

```
  s(sub-region,
    bs = 'mrf',
    xt = list(nb = data$nb),
    k = k) +                    # ICAR varying intercept at level sub-region
  s(sub-region, by = covariate,
    bs = 'mrf',
    xt = list(nb = data$nb),
    k = k),           # ICAR varying slope for covariate at level sub-region
data = data,
method = "REML")
```

When labelling the new columns which are augmented to the original dataframe from such a model, `st_augment()` follows the formula syntax of the **lme4** (Bates et al. 2015) package, where the *pipe symbol* (`|`) indicates "*grouped by*". Table 3 shows how the augmented columns in this scenario would be labelled. Each column name begins with either `random.effect.` or `mrf.smooth.` as appropriate. An additional column is also added for the standard error of each estimate. These columns are named as above but with `se.` prepended (e.g. `se.random.effect.region`).

st_quickmap_preds()

These estimates can then be quickly mapped. As it is possible to include more than 1 varying component, the output of this function is a list of plots. They can be viewed individually by indexing, or all at one using the `plotlist=` argument from the `ggarrange()` function which is part of the **ggpubr** (Kassambara 2023) package. We will see examples of this in the following examples. The maps are automatically titled and subtitled according to the type of effect. For example, the map showing `random.effect.region` will have "*region*" as its title and "*random.effect*" as its subtitle.

5 Example 1: Indonesia

Modelling earthquakes in Indonesia seems like a good example to demonstrate this package. Firstly, Indonesia is composed of many islands. Secondly, earthquake activity is known to be associated with the presence of faults which exist below sea level and thus do not respect land boundaries. Therefore it is reasonable to expect similar behaviour in nearby provinces regardless of whether or not they are contiguous. We aim to model the intensity of earthquake activity by province across Indonesia, controlling for proximity to faults.

Data

The data for this section have been taken from the USGS earthquake catalogue API. The datasets and an explanation are available at `https://github.com/horankev/quake_data` and record all earthquakes in and close to Indonesia from 1985-2023. Below is a map of Indonesia, with other neighbouring or bordering countries filled in grey. The many local faults which lie within 300km of the shore are shown as green lines in Figure 6.

To get an interpretable measure of the concentration of faults in any area they are transformed from linestrings to polygons by setting a buffer of 10km around them, as shown below. Now both our faults and the sizes of provinces are in units of kilometres squared. This means we can generate a unitless metric of what proportion of any administrative unit is covered by these buffered faults. This measure across provinces is shown in Figure 7.

Earthquake activity will be seen as the total number of earthquakes with their epicentre in each province per unit area of that province. We have restricted counts to earthquakes of magnitude >5.5 which is the point at which they are often labelled as damaging.

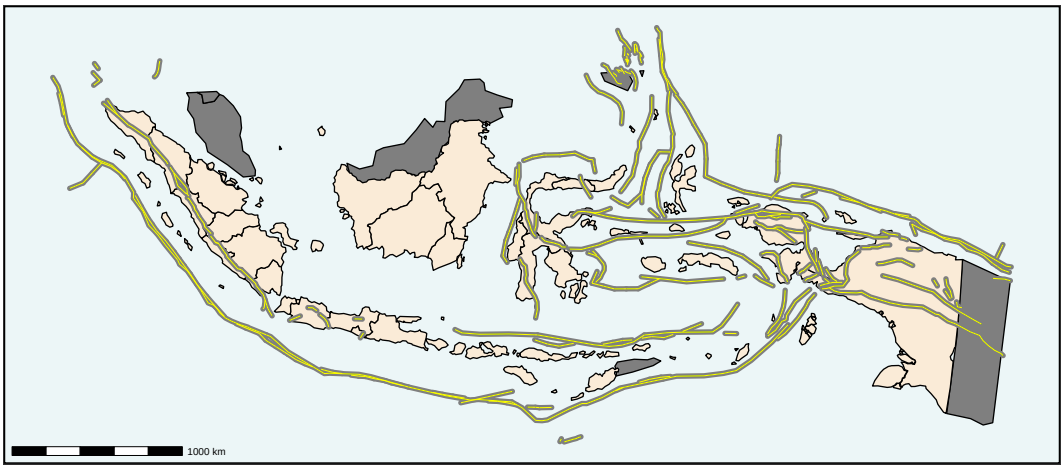

Figure 6: Indonesia faults. Surrounded by a 10 kilometre buffer.

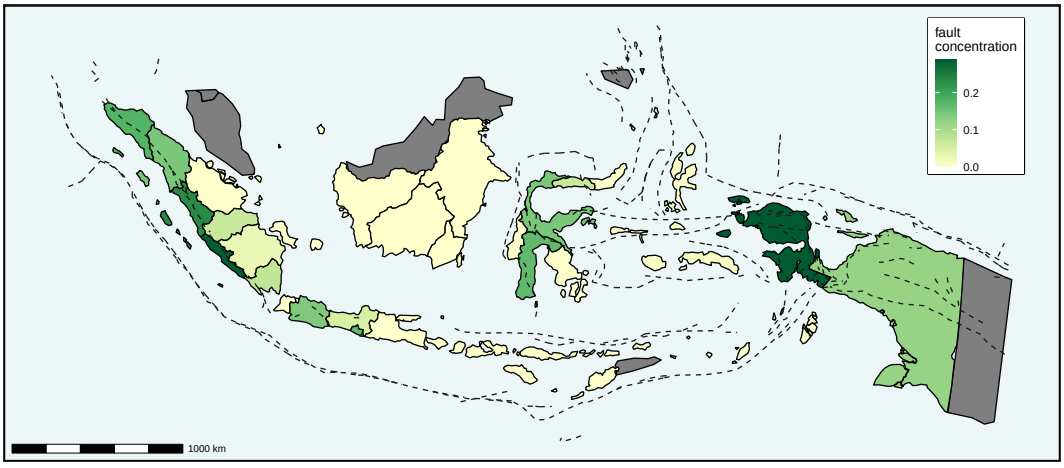

Figure 7: Indonesia fault concentration. Square kilometre of buffered fault per square kilometre of province area.

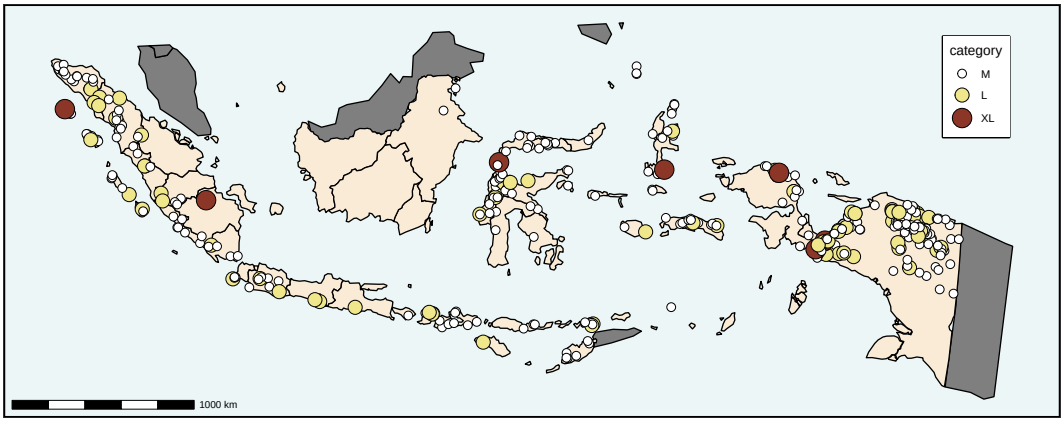

Figure 8: Earthquakes in Indonesia, 1985-2023. Categorised as medium, large or extra-large.

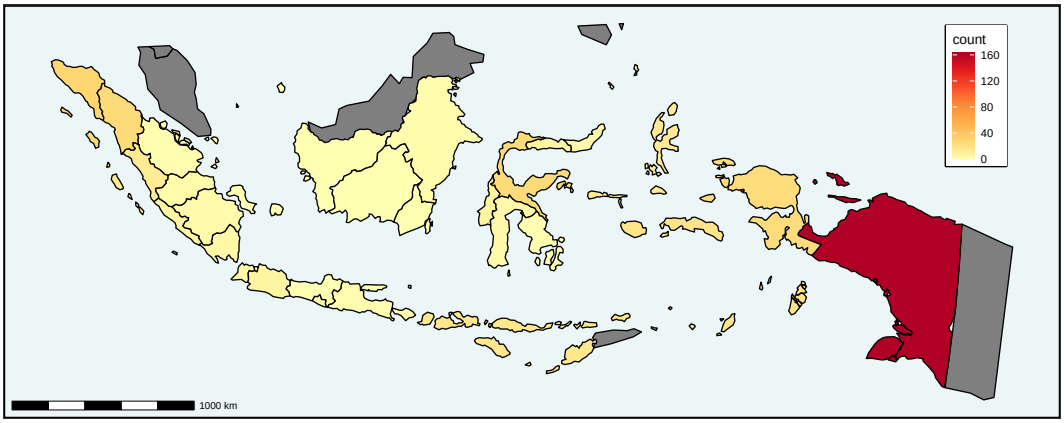

Figure 9: Earthquake count in Indonesia, 1985-2023, mag > 5.5: count by province.

The occurrences of these earthquakes are shown in Figure 8, their total per province in Figure 9, and finally, their count per square kilometre can be seen in Figure 10.

Model

As this is count data, we will model it as a poisson distribution with λ as the mean count per province. For $i = 1, ..., n$ provinces, the dependent variable in this model is

$$y_i = \text{earthquake incidence}_i = \frac{\text{earthquake count in province}_i}{\text{province area}_i}, \tag{1}$$

while the explanatory variable is

$$x_i = \text{fault intensity}_i = \frac{\text{area of buffered faults in province}_i}{\text{province area}_i}. \tag{2}$$

Firstly, when excluding the incidence and just modelling counts, where $y_i = \text{earthquake count in province}_i$, the poisson model is of the following form:

$$y_i|\lambda_i \sim \text{Pois}(\lambda_i) \tag{3}$$

with

$$E(y_i|\lambda_i) = \lambda_i. \tag{4}$$

We model

$$log(\lambda_i) = \beta_0 + \beta_1 x_i + \gamma_i. \tag{5}$$

where γ_i is a term with a correlation structure reflecting a province's location relative to other provinces.

We can describe these relationships by setting up a neighbourhood structure based on queen contiguity where a pair of provinces are considered neighbours if they share at least one point of boundary. This can be modelled as a Markov random field to generate an ICAR model with a spatially

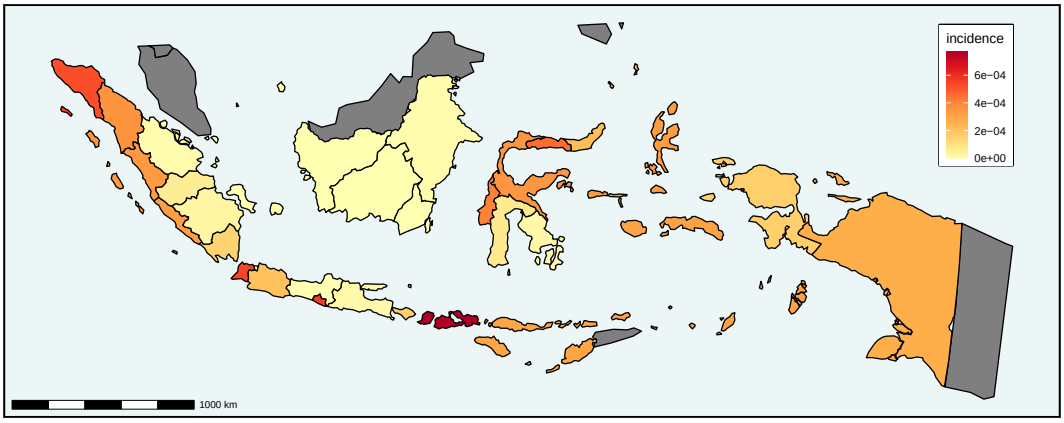

Figure 10: Earthquake incidence in Indonesia, 1985-2023, mag > 5.5: count per square kilometre by province.

varying term. Each of these terms will be correlated the others according to the neighbourhood structure we have defined.

The Markov random field here follows a multivariate Gaussian distribution. γ_i is a vector of province effects having a distribution with mean **0** and precision **P** where

$[\mathbf{p}]_{ij} = v_i$ if $i = j$ and v_i is the number of adjacent provinces to province i,

$[\mathbf{p}]_{ij} = -1$ if provinces i and j are adjacent, and

$[\mathbf{p}]_{ij} = 0$ otherwise.

A further constraint that $\Sigma_j \gamma_j = 0$ is applied so that the distribution is identifiable.

If we include an offset term (here, area) because we are more interested in the incidence than in the actual count, then

$$log(\frac{\lambda_i}{\text{area}_i}) = \beta_0 + \beta_1 x_i + \gamma_i \tag{6}$$

which is equivalent to

$$log(\lambda_i) = \beta_0 + \beta_1 x_i + \gamma_i + log(\text{area}_i). \tag{7}$$

We are still modelling $log(\lambda)$ rather than the incidence, but we are adding an offset to adjust for differing areas. Modelling $log(\lambda)$ and adding an offset is equivalent to modelling incidence, and coefficients can be interpreted that way.

When interpreting the estimated coefficients of the model, it can be useful to look at it in the following form:

$$\lambda_i = e^{\beta_0 + \beta_1 x_i + \gamma_i} \text{area}_i. \tag{8}$$

Pre-functions

Such models, however, can not deal with locations which have no neighbours. In the case of Indonesia, this is quite problematic. It is composed of many islands. The estimated count of islands according to Andréfouët, Paul, and Farhan (2022) is 13,558. While it is not unusual for a country to have a number of often small offshore islands, Indonesia is entirely composed of (at least portions of) an archipelago of islands, so many of these islands or groups of islands are individual provinces in their own right. We might like them to be associated with their k nearest neighbours. However, a nearest neighbours approach to Indonesia as a whole might not be desired. Therefore, they must either be excluded from the study or somehow brought within the neighbourhood framework.

In this case, we use `st_bridges()` for setting up the queen contiguity structure as usual, but with the additional stipulation that unconnected units (provinces which are islands or collections of islands) are considered neighbours to their k nearest provinces. For this example, we have set the value of k to 2. The resulting neighbourhood structure is shown in Figure 11.

```
# join islands to k=2 nearest neighbours
# various arguments exist for altering colours and sizes
# additional ggplot themes and layers can be added

st_bridges(provinces_df, "province",link_islands_k = 2) |>
  st_quickmap_nb(fillcol = "antiquewhite1",
```

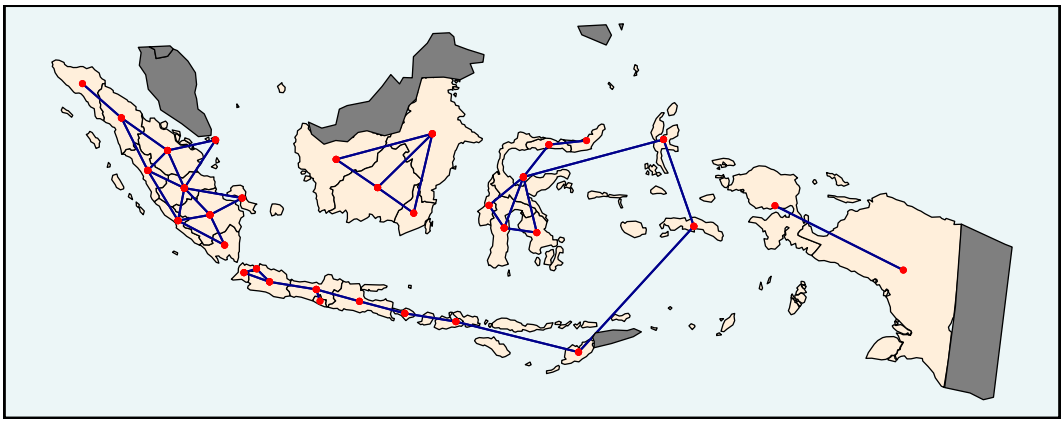

Figure 11: Neighbourhood structure for Indonesian provinces created by `st_bridges()` with `k=2`.

```
                bordercol = "black", bordersize = 0.5,
                linkcol = "darkblue", linksize = 0.8,
                pointcol = "red", pointsize = 2) +
  theme(panel.background = element_rect(fill = "#ECF6F7", colour = "black",
                                        linewidth=1.5),
        axis.text = element_blank()) +
  geom_sf(data=nearby_countries_df,
          fill="gray50", linewidth=0.5, colour="black")
```

This neighbourhood structure now has no unconnected provinces so it is suitable for use in an ICAR model. However, we will

- add some additional contiguities using `st_manual_join_nb()`
- and remove one using `st_manual_cut_nb()`.

To cater for the possibility that a modeller might not be familiar with the names of the various geographic units but still wishes to make manual alterations to their relationships, we can look at a map (Figure 12) where the nodes are shown by index number instead of as points (using the argument `nodes='numeric'`). This makes it easy to manually cut and join neighbours as desired. Furthermore, there is an option to show concave hulls drawn around each unit (using `concavehull = TRUE`). This is also shown in Figure 12. These shapes are not used in the assignment of contiguities but it can be useful to see them in a situation such as Indonesia where many individual provinces are actually multipolygons. Without them, it is not clear whether an island is a province in its own right, or which group of islands together form one province.

```
# with 'concavehull = TRUE' and 'nodes = "numeric"'

st_bridges(provinces_df, "province",link_islands_k = 2) |>
  st_quickmap_nb(fillcol = "antiquewhite1",
                 bordercol = "black", bordersize = 0.5,
                 linkcol = "tomato", linksize = 0.5,
                 nodes = "numeric",
                 numericcol = "black", numericsize = 6,
                 concavehull = TRUE,
                 hullcol = "darkgreen", hullsize = 0.2) +
  theme(panel.background = element_rect(fill = "#ECF6F7", colour = "black",
                                        linewidth=1.5),
        axis.text = element_blank())


# with 'concavehull = TRUE' and 'nodes = "numeric"'

st_bridges(provinces_df, "province",link_islands_k = 2) |>
  st_quickmap_nb(fillcol = "antiquewhite1",
                 bordercol = "black", bordersize = 0.5,
                 linkcol = "tomato", linksize = 0.5,
```

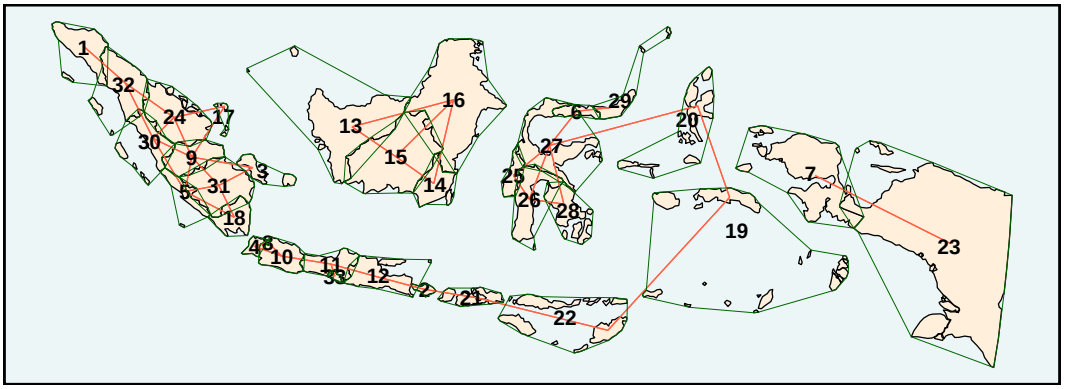

Figure 12: Neighbourhood structure for Indonesian provinces. Viewed with `st_quickmap_nb()`, using the arguments `nodes = 'numeric'` and `concavehull = TRUE`.

```
                nodes = "numeric",
                numericcol = "black", numericsize = 6,
                concavehull = TRUE,
                hullcol = "darkgreen", hullsize = 0.2) +
  theme(panel.background = element_rect(fill = "#ECF6F7", colour = "black",
                                        linewidth=1.5),
        axis.text = element_blank())
```

Having made some manual adjustments to the neighbourhood structure, the new structure can be seen in Figure 13. Edge effects have also been mitigated by additionally connecting the two extreme provinces (1 and 23), which would otherwise have only one neighbour, to their two next closest neighbours.

```
# a series of manual joins and cuts by index number

st_bridges(provinces_df, "province",link_islands_k = 2) |>
  st_manual_join_nb(1,24) |>
  st_manual_join_nb(1,30) |>
  st_manual_join_nb(3,13) |>
  st_manual_join_nb(13,17) |>
  st_manual_join_nb(14,25) |>
  st_manual_join_nb(20,29) |>
  st_manual_join_nb(19,23) |>
  st_manual_join_nb(16,27) |>
  st_manual_join_nb(22,23) |>
  st_manual_join_nb(7,19) |>
  st_manual_join_nb(7,20) |>
  st_manual_join_nb(19,28) |>
  st_manual_join_nb(4,18) |>
  st_manual_join_nb(21,26) |>
  st_manual_join_nb(22,28) |>
  st_manual_cut_nb(19,22) |>
  st_quickmap_nb(fillcol = "antiquewhite1",
                bordercol = "black", bordersize = 0.5,
                linkcol = "darkblue", linksize = 0.8,
                pointcol = "red", pointsize = 2) +
  theme(panel.background = element_rect(fill = "#ECF6F7", colour = "black",
                                        linewidth=1.5),
        axis.text = element_blank()) +
  geom_sf(data=nearby_countries_df,
          fill="gray50", linewidth=0.5, colour="black") +
  annotation_scale()
```

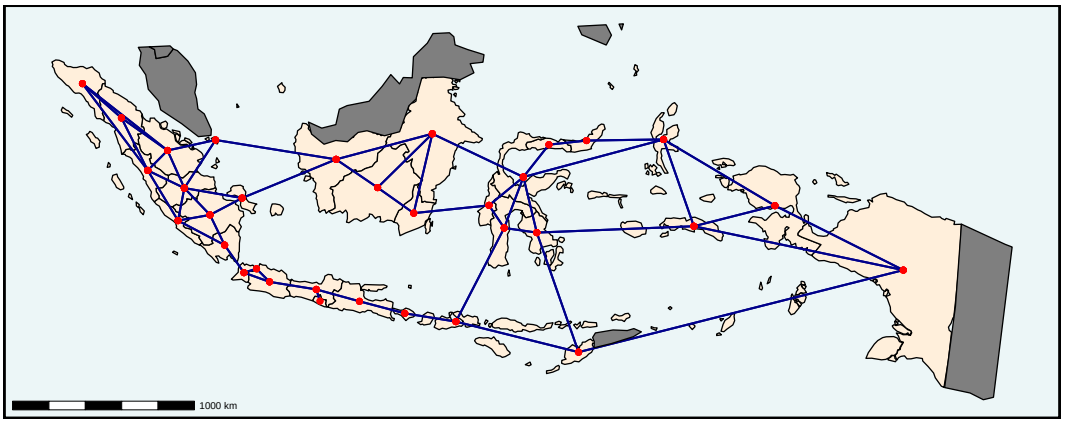

Figure 13: Neighbourhood structure for Indonesian provinces, after alterations using `st_manual_join()` and `st_manual_cut()`.

`mgcv` model

We now create the ICAR model using **mgcv**. We will be able to simply pass the contiguity structure we have created as both the data and a column of it as the neighbourhood structure.

```
mod_pois_mrf <- gam(damaging_quakes_total ~
                      fault_concentration +
                      s(province, bs='mrf', xt=list(nb=prep_data$nb), k=22) +
                      offset(log(area_province)),
                    data=prep_data, method="REML",family = "poisson")
```

The adjusted R-squared is **0.957** and deviance explained is **88.2%**. The estimated coefficient associated with fault concentration is -3.2355. As this is positive, it is greater than 1 when exponentiated:

$$\exp(0.7531) = 0.04$$

and this is the expected multiplicative association between a unit change in fault concentration and incidence of earthquakes in a province.

Controlling for this estimated mean elevated risk of earthquakes due to concentration of faults, what is the additional risk level of earthquakes in a province? This can be seen as a measure of the activity level of faults locally and it is spatially smoothed by the autoregressive process. It is mapped as the ICAR component of the model below:

Post-functions

Use `st_augment()` to get estimates from **mgcv** model in format similar to that used by the **broom** package. For instance, we see from the code below that the original dataframe is now augmented with columns for "mrf.smooth.province" and "mrf.smooth.province" which shows the estimate and standard errors for the γ_i component. Note that this is how we would expect them to be named, based on the previous discussion surrounding Table 3.

```
mod_pois_mrf |>
  st_augment(prep_data) |>
  names()
```

```
#>  [1] "S"                     "M"
#>  [3] "L"                     "XL"
#>  [5] "quake_total"           "quake_density"
#>  [7] "damaging_quakes_total" "damaging_quakes_density"
#>  [9] "area_fault_within"     "area_province"
#> [11] "fault_concentration"   "nb"
#> [13] "mrf.smooth.province"   "se.mrf.smooth.province"
#> [15] "province"              "province_id"
#> [17] "geometry"
```

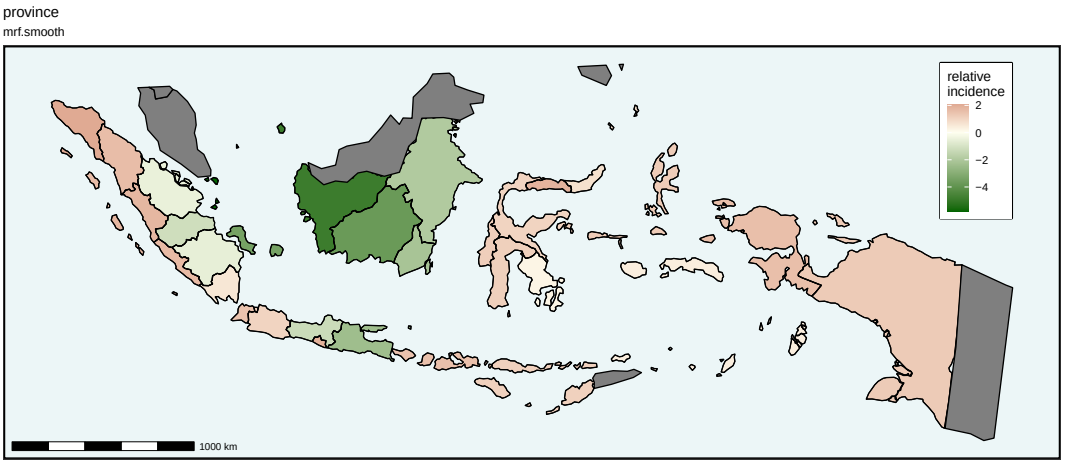

Figure 14: Estimates of γ_i shown as a map using `st_quickmap_preds()`.

Use `st_augment()` and `st_quickmap_preds` in the following code to get a quick visualisation of the estimates for γ_i, as shown in Figure 14. Again, note that the title and subtitle of the image are as previously discussed.

```
# st_quickmap_preds() outputs a list of ggplots

plot_mrf <- mod_pois_mrf |>
  st_augment(prep_data) |>
  st_quickmap_preds(scale_low = "darkgreen",
                    scale_mid = "ivory",
                    scale_high = "darkred",
                    scale_midpoint = 0)

# in this case, there is only one plot in the list
# so we call it by index

plot_mrf[[1]] +
  coord_sf(datum=NA) +
  theme(panel.background = element_rect(fill = "#ECF6F7", colour = "black",
                                        linewidth=1.5),
        axis.text = element_blank()) +
  geom_sf(data=provinces_df, fill=NA, colour="black", linewidth=0.5) +
  geom_sf(data=nearby_countries_df, fill="gray50", colour="black",
          linewidth=0.5) +
  labs(fill="relative\nincidence") +
  annotation_scale() +
  coord_sf(datum=NA) +
  theme(legend.position = c(0.92,0.77),
        legend.box.background = element_rect(colour = "black", linewidth = 1),
        legend.title = element_text())
```

If we wish to apply the inverse link function (the exponential function in the case of this poisson model) to map these values to a more interpretable scale, this will not be generated by the function `st_quickmap_preds()`. Instead, we must use the augmented dataframe which is produced by `st_augment()` and create the appropriate extra column with the usual **tidyverse** `mutate()` function. This allows us to produce the map in Figure 15. As these coefficients are multiplicatively related to the earthquake incidence, values below 1 imply an earthquake incidence which is lower than expected.

The provinces with the 3 most elevated incidences are labelled in red We can see that, controlling for the added risk of proximity to faults, the province of Aceh has 7.9 times the expected incidence, or number of major earthquakes per square kilometre. The two lowest-scoring provinces, labelled in green, have essentially no incidence of earthquake epicentres, controlling for what their proximity to faults alone would suggest.

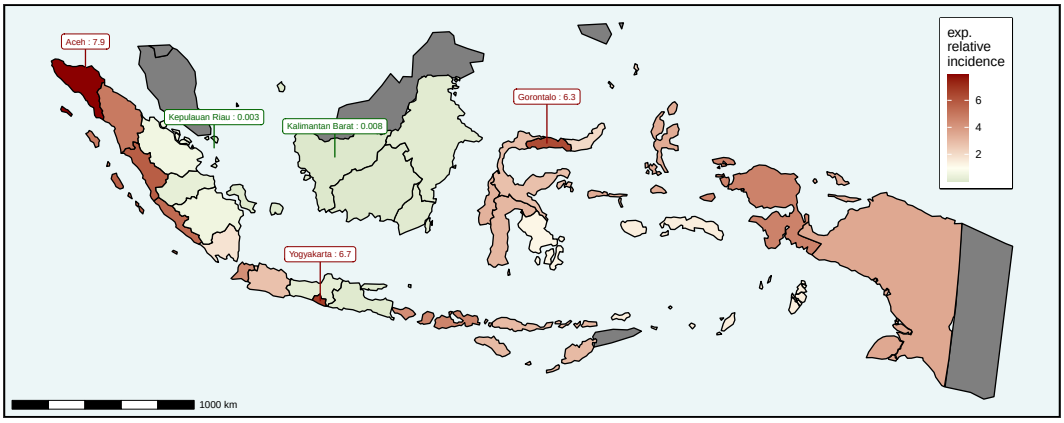

Figure 15: Map showing estimates of $\exp(\gamma_i)$. This is produced by adding an additional column to the dataframe produced by `st_augment()`.

Workflow

In this example, we have gone through a number of stages carefully, making changes to contiguities that we deemed appropriate as we went. However, in practice, at least in a first iteration, it might not be necessary to go through all of these steps. A rough and ready model (and visual output) can be generated with `sfislands` using nothing more than three or four lines of code, such as the following:

```
# set up neighbourhood structure

prep_data <- st_bridges(provinces_df, "province")

# define model

mod <- gam(quake_mlxl_total ~
                  fault_concentration +
                  s(province, bs='mrf', xt=list(nb=prep_data$nb), k=22) +
                  offset(log(area_province)),
        data=prep_data, method="REML",family = "poisson")

# augment tidy estimates

tidy_ests <- st_augment(mod, prep_data)

# visualise them

st_quickmap_preds(tidy_ests)
```

6 Example 2: London

The next example looks only at using the *pre-functions*, but in a situation where the presence of islands is not the issue. Consider the wards and boroughs of London (sourced from the Greater London Authority's London Datastore). In Figure 16 the `st_bridges()` function is applied to them to construct a queen contiguity neighbourhood structure. Because there are no disjoint units (or "islands"), this will be the same as using `st_contiguity()` from **sfdep**. The `st_quickmap_nb()` function gives an immediate visual representation of the structure. This can be supplemented by adding a layer showing the course of the river Thames.

```
# same as sfdep:st_contiguity() as there are no islands
# an extra layer for the river Thames

st_bridges(london, "NAME") |>
  st_quickmap_nb() +
  geom_sf(data=thames, colour="blue", linewidth=1.5) +
  theme(panel.background = element_rect(fill = "#F6F3E9", colour = "black",
```

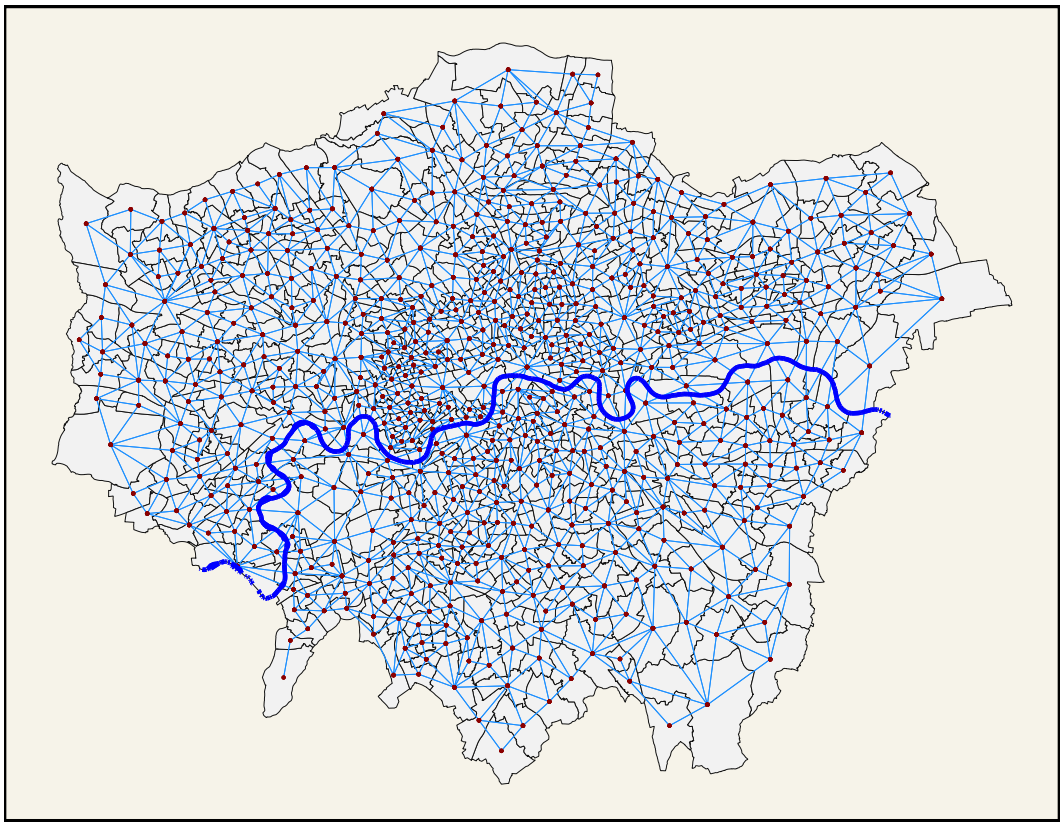

Figure 16: Wards and boroughs of Greater London. Queen contiguity.

```
                                        linewidth=1.5))
```

When the study area has a river running through it there can be problems with constructing neighbourhood structures. Depending on how the geometries are defined, the presence of a river can either mean that no units on either side of its banks are considered neighbours or, at the other extreme, the river is essentially ignored and all units on opposing banks are considered neighbours.

Depending on the presence of river crossings, two areas which are physically quite close might be very distinct. Somebody living on the banks of a river might be more likely to go about their life primarily on their side of the river, despite the short distance as the crow flies of facilities on the other side.

`sfislands` makes it easier to take account of this situation. We can look at just those boroughs which are on either side of the river Thames (Figure 17)

```
# which boroughs are alongside the river

riverside <- thames |> st_intersects(london) |> unlist() |> unique()

# only map these boroughs

st_bridges(london[riverside,],"NAME") |>
  st_quickmap_nb(linksize = 0.5) +
  geom_sf(data=thames, colour="blue", linewidth=1.5) +
  annotation_scale(location="br") +
  coord_sf(datum=NA) +
  theme(panel.background = element_rect(fill = "#F6F3E9", colour = "black",
                                        linewidth=1.5))
```

In order to take account of actual connectivity, we can add a layer showing the road and pedestrian bridges or tunnels. These were scraped from a Wikipedia article describing the various crossings of the river Thames. In Figure 18, we have also drawn a 1 kilometre buffer around them. Until we get to the eastern portion of the map, all units on opposing sides of the river are within 1 kilometre of a crossing. This is not the case for the extreme eastern units so we might not want to consider these units neighbours if the river flows between them.

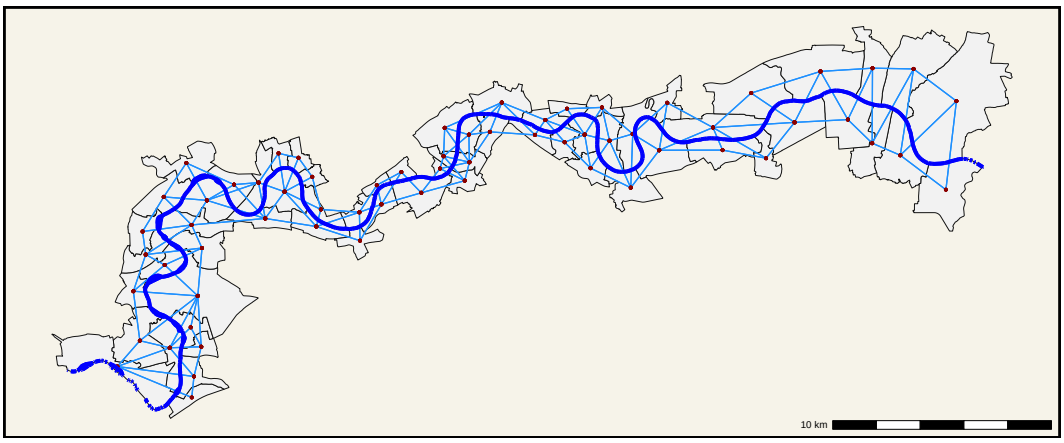

Figure 17: Riverside boroughs of Greater London. Queen contiguity disregarding river.

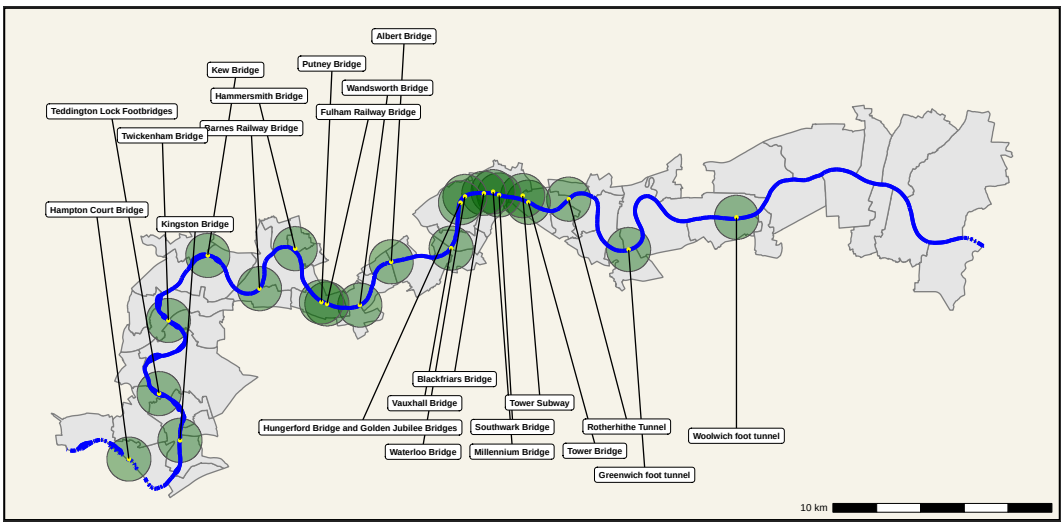

Figure 18: Riverside boroughs of Greater London. Road and pedestrian crossing and tunnels surrounded by 1 kilometre buffer shaded green.

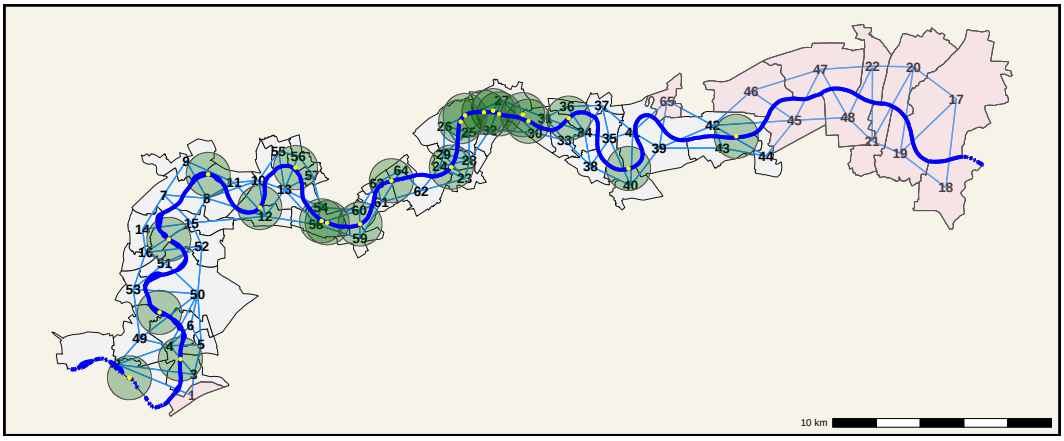

Figure 19: Riverside boroughs of Greater London. Index number for each borough shown at centroid. Boroughs which are not within 1 kilometre of a crossing are shaded pink.

In order the identify the changes we wish to make, we use the `nodes = "numeric"` argument in `st_quickmap_nb()`. Now we can identify each unit by its position in the contiguity structure. Here we have shaded in pink the units which are not within 1 kilometre of a river crossing (see Figure 19).

```
# with 'nodes = "numeric"'

st_bridges(london[riverside,],"NAME") |>
  st_quickmap_nb(nodes = "numeric", numericsize = 4, linksize = 0.5) +
  geom_sf(data=no_touch_buffer, fill="pink", alpha=0.3) +
  geom_sf(data=crossings_roadped |> st_buffer(1000),
          fill="darkgreen", alpha=0.3) +
  geom_sf(data=thames, colour="blue", linewidth=1.5) +
  geom_sf(data=crossings_roadped, size=1, colour="yellow") +
  annotation_scale(location="br") +
  coord_sf(datum=NA) +
  theme(panel.background = element_rect(fill = "#F6F3E9", colour = "black",
                                        linewidth=1.5))
```

This allows us to easily cut the ties across the river for these units manually by using the function `st_manual_cut_nb()`. The results of this editing procedure are shown in Figure 20. While we are using the index of the units in this example, the function also accepts names as arguments which may be more convenient in some circumstances. Having made these adjustments, `st_quickmap_nb()` now shows a connectivity structure (Figure 20) which reflects the influence of the presence or absence of river crossings.

This example shows that the pre-functions of `sfislands` have uses for situations which do not involve islands. They can be used to apply domain knowledge to easily design the most appropriate neighbourhood structure.

```
# manually cut the links where there is no crossing

st_bridges(london[riverside,],"NAME") |>
  st_manual_cut_nb(18,17) |>
  st_manual_cut_nb(19,17) |>
  st_manual_cut_nb(19,20) |>
  st_manual_cut_nb(20,21) |>
  st_manual_cut_nb(21,22) |>
  st_manual_cut_nb(22,48) |>
  st_manual_cut_nb(47,48) |>
  st_manual_cut_nb(45,46) |>
  st_manual_cut_nb(45,47) |>
  st_manual_cut_nb(39,65) |>
  st_manual_cut_nb(1,2) |>
  st_quickmap_nb(bordercol = "black", bordersize = 0.5, linksize = 0.5) +
  geom_sf(data=no_touch_buffer, fill="pink", alpha=0.3) +
```

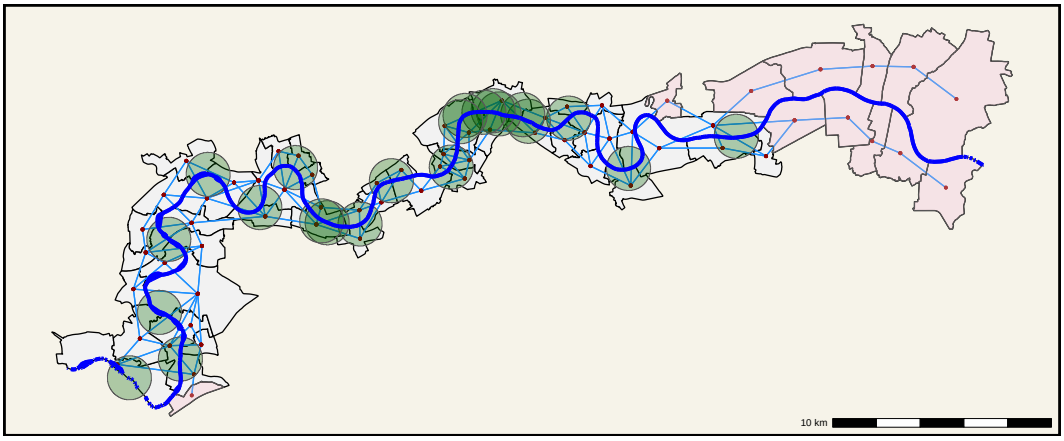

Figure 20: Riverside boroughs of Greater London. Contiguities across the river have been cut for the pink boroughs.

Table 4: Three nested levels of administrative divisions in Liverpool.

code	area type	description
OA	output areas	lowest level of geographical area for census statistics, usually containing 100 - 625 persons
LSOA	lower layer super output areas	usually 4 or 5 OAs
MSOA	middle layer super output areas	usually 4 or 5 LSOAs

```
  geom_sf(data=crossings_roadped |> st_buffer(1000),
          fill="darkgreen", alpha=0.3) +
  geom_sf(data=thames, colour="blue", linewidth=1.5) +
  annotation_scale(location="br") +
  coord_sf(datum=NA) +
  theme(panel.background = element_rect(fill = "#F6F3E9", colour = "black",
                                        linewidth=1.5))
```

7 Example 3: Liverpool

In this final example, there are no islands or river issues. We saw in the Indonesia example how `sfislands` can extract estimates from the `mgcv` argument `bs="mrf"`. It can also deal with random effects as defined by `bs="re"` in `mgcv` models. Here, we will construct a hierarchical model in this way which also features an ICAR component. We will compare two different types of contiguity which could be applied:

- `st_bridges()` from `sfislands`, and
- `st_knn()` from **sfdep**.

We will show how their outcomes can be quickly mapped to give an immediate sense of the differences.

```
# magnify a section

from <- c(xmin = -2.97, xmax = -2.95, ymin = 53.38, ymax = 53.39)
to <- c(-3.01,-2.96,53.32,53.35)

st_bridges(liverpool, "oa_cd") |>
  st_quickmap_nb(pointsize = 0.1,
                 linksize = 0.1,
                 fillcol = "gray90") +
  geom_sf(data=mersey, fill="lightblue", colour="gray30",
          linewidth=0.2, alpha=0.5) +
  geom_magnify(data=liverpool,
               from = from, to = to,
```

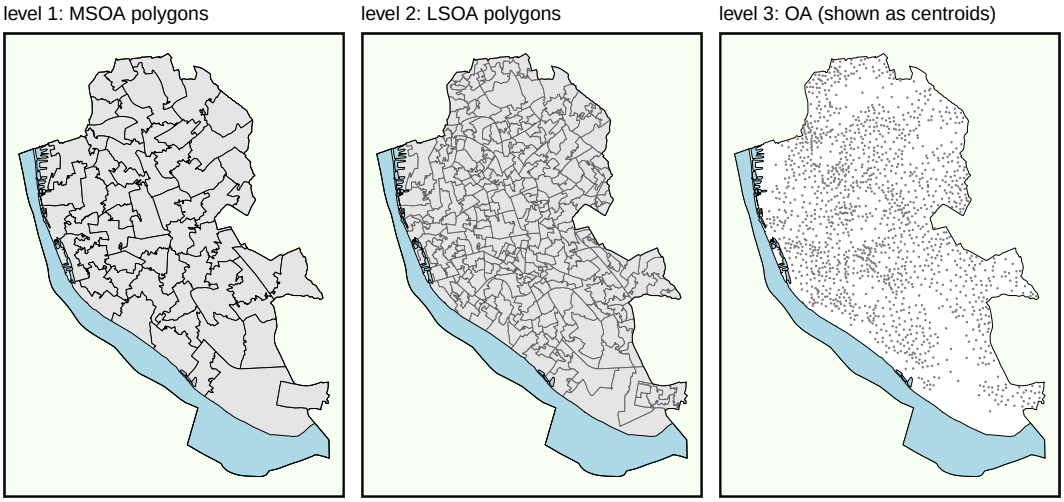

Figure 21: Nested levels of administrative divisions in Liverpool.

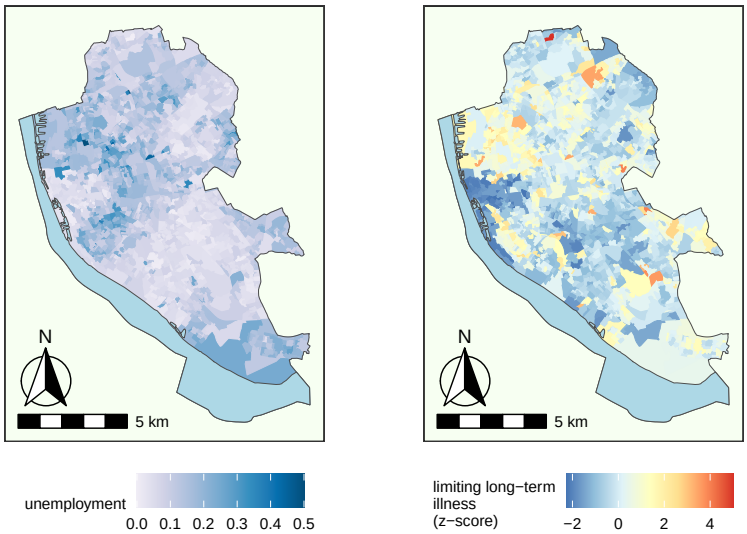

Figure 22: Unemployment and limiting long-term illness rates in Liverpool, with extent of Mersey estuary in light blue to the south west.

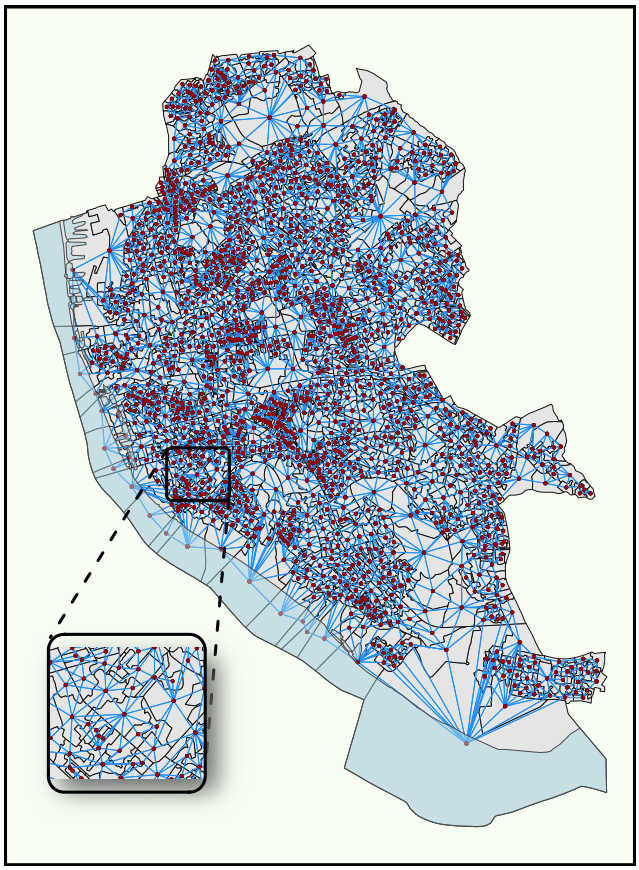

Figure 23: First order queen contiguity of output areas (OAs) in Liverpool with magnified inset.

```
            corners = 0.1, shadow = TRUE, linewidth = 0.6) +
theme(panel.background = element_rect(fill = "#F7FFF2",
                                      colour = "black", linewidth=1))
```

This example is based on data from the Office of National Statistics as presented by Rowe and Arribas-Bel (2024) in their demonstration of techniques for hierarchical spatial regression. They use the different levels of nested administrative boundaries in Liverpool for their demonstration. These are described in Table 4, and are mapped separately in Figure 21.

Now suppose we wish to model unemployment as a function of limited long-term illness, taking into account this hierarchical structure. Limiting long-term illness (LLTI) refers to a health problem or disability which limits a person's day-to-day activities (Barnett 2001). We wish to incorporate a fixed intercept and coefficient for LLTI, with additional random intercepts and slopes at MSOA and LSOA levels. The distribution of these variables across the OAs of Liverpool are shown in Figure 22.

If y_{ijk} and x_{ijk} are respectively the levels of unemployment and LLTI in each of i OAs within j LSOAs, which in turn are within k MSOAs, then

$$y_{ijk} = \beta_{0jk} + \beta_{1jk}x_{ijk} + \gamma_{ijk} + \epsilon_{ijk} \tag{9}$$

$$\beta_{0jk} = \beta_{0k} + u_{0jk} \tag{10}$$

$$\beta_{1jk} = \beta_{1k} + u_{1jk} \tag{11}$$

$$\beta_{0k} = \beta_0 + w_{0k} \tag{12}$$

$$\beta_{1k} = \beta_1 + w_{1k} \tag{13}$$

$$\epsilon_{ijk} \sim N(0, \sigma_\epsilon^2) \tag{14}$$

$$u_{jk} \sim N(0, \sigma_u^2) \tag{15}$$

$$w_k \sim N(0, \sigma_w^2) \tag{16}$$

where the γ_{ijk}s comprise a vector of ICAR components for each OA, similar to the provinces in the earlier Indonesia model. The vector has a multivariate normal distribution with mean 0 and precision related to the contiguity structure as before.

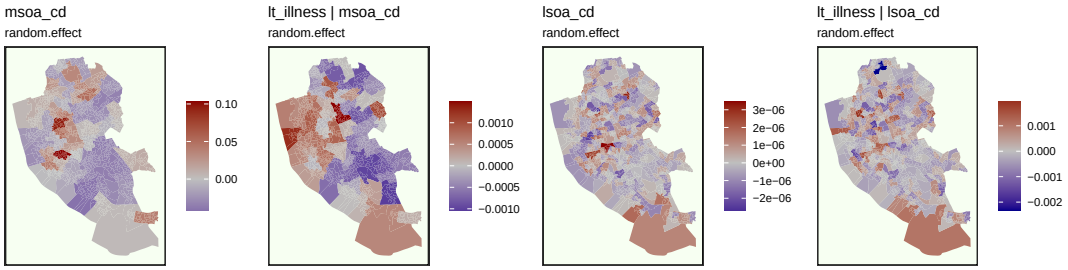

Figure 24: Random intercept and slope estimates from hierarchical model. Middle and lower layer super output areas as levels.

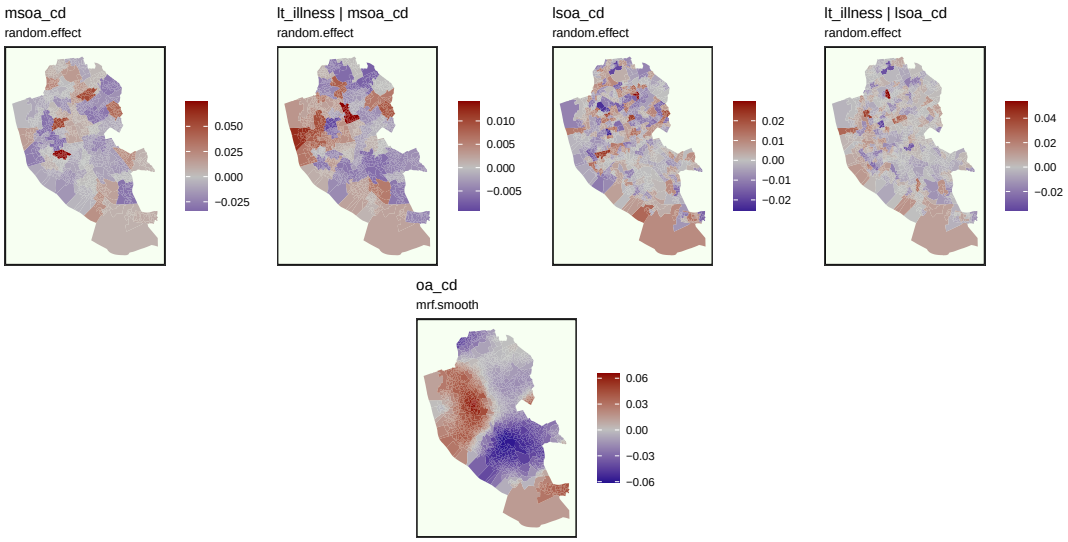

Figure 25: Random intercept and slope estimates from hierarchical model. Middle and lower layer super output areas are the top two levels respectively (MSOA & LSOA). Output areas (OA) as lowest level Markov random field smooth using first-degree queen contiguity from `st_bridges()`.

There are no islands or discontiguities in this geometry of Liverpool so using `st_bridges()` will give the same structure as would `sfdep::st_contiguity()` (see Figure 23).

The hierarchical model with no mrf component produces estimates as shown in Figure 24.

If we instead use contiguity as the basis of an additional spatial term, we get estimates shown in Figure 25.

If instead we consider units to be neighbours if they lie within a distance band of 700 metres, we get estimates shown in Figure 26.

A comparison of their AICs (see Table 5) suggests that, in this scenario, queen contiguity is more appropriate than the distance band condition.

8 Summary

These examples have shown the varying scenarios in which `sfislands` can be useful. It aims to contribute to spatial modelling by making an awkward area less awkward. Rather than having a default attitude of ignoring islands when building neighbourhood structures based on contiguity, it hopes to encourage at least an examination of whether or not it is appropriate for them to be included. It also provides helper functions to use these structures in spatial regression models built with `mgcv`

Table 5: Comparison of models by AIC.

model	AIC	relative AIC
hierachical model, ICAR based on st_bridges()	-5126.948	-1.000
hierachical model, ICAR based on sfdep::st_dist_band()	-5107.790	-0.996
hierachical model, no ICAR	-5082.684	-0.991

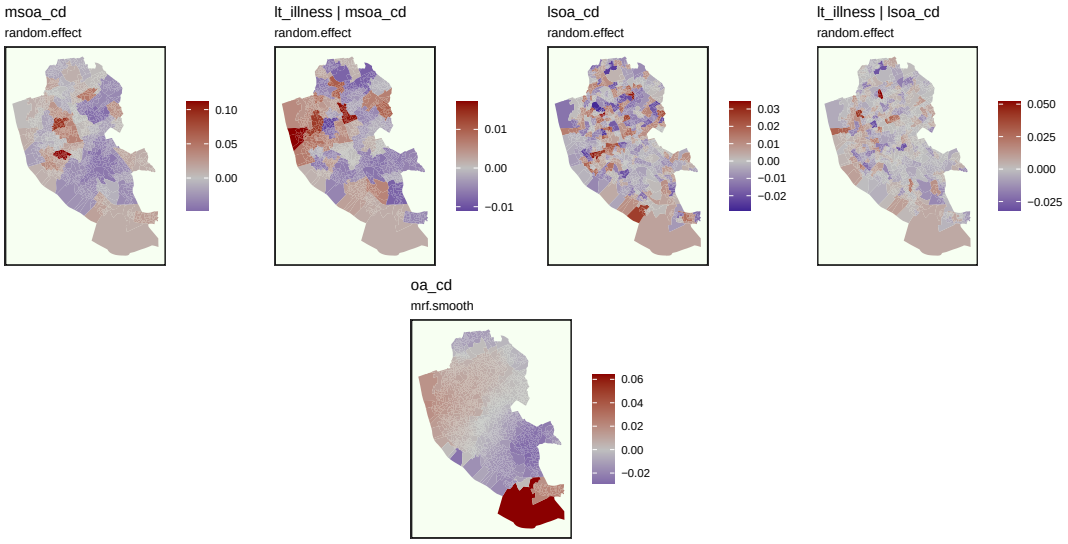

Figure 26: Random intercept and slope estimates from hierarchical model. Middle and lower layer super output areas are the top two levels respectively (MSOA & LSOA). Output areas (OA) as lowest level Markov random field smooth using `sfdep::st_dist_band()` with radius of 700 metres.

which streamline the human effort necessary to examine the estimates.

9 Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 18/CRT/6049.

References

Andréfouët, Serge, Mégane Paul, and A. Riza Farhan. 2022. "Indonesia's 13558 Islands: A New Census from Space and a First Step Towards a One Map for Small Islands Policy." *Marine Policy* 135 (January): 104848. https://doi.org/10.1016/j.marpol.2021.104848.

Bakka, Haakon, Håvard Rue, Geir-Arne Fuglstad, Andrea Riebler, David Bolin, Elias Krainski, Daniel Simpson, and Finn Lindgren. 2018. "Spatial Modelling with r-INLA: A Review." arXiv. https://doi.org/10.48550/ARXIV.1802.06350.

Barnett, S. 2001. "A Multilevel Analysis of the Effects of Rurality and Social Deprivation on Premature Limiting Long Term Illness." *Journal of Epidemiology &Amp; Community Health* 55 (1): 44–51. https://doi.org/10.1136/jech.55.1.44.

Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. "Fitting Linear Mixed-Effects Models Using lme4." *Journal of Statistical Software* 67 (1): 1–48. https://doi.org/10.18637/jss.v067.i01.

Bivand, Roger, Andrew Bernat, Marilia Carvalho, Yongwan Chun, Carsten Dormann, Stéphane Dray, Rein Halbersma, et al. 2005. "The Spdep Package." *Comprehensive R Archive Network, Version*, 05–83.

Briz-Redón, Álvaro, Adina Iftimi, Juan Francisco Correcher, Jose De Andrés, Manuel Lozano, and Carolina Romero-García. 2021. "A Comparison of Multiple Neighborhood Matrix Specifications for Spatio-Temporal Model Fitting: A Case Study on COVID-19 Data." *Stochastic Environmental Research and Risk Assessment* 36 (1): 271–82. https://doi.org/10.1007/s00477-021-02077-y.

Bürkner, Paul-Christian. 2017. "brms: An R Package for Bayesian Multilevel Models Using Stan." *Journal of Statistical Software* 80 (1): 1–28. https://doi.org/10.18637/jss.v080.i01.

Duncan, Earl W., Nicole M. White, and Kerrie Mengersen. 2017. "Spatial Smoothing in Bayesian Models: A Comparison of Weights Matrix Specifications and Their Impact on Inference." *International Journal of Health Geographics* 16 (1). https://doi.org/10.1186/s12942-017-0120-x.

Earnest, Arul, Geoff Morgan, Kerrie Mengersen, Louise Ryan, Richard Summerhayes, and John Beard. 2007. "Evaluating the Effect of Neighbourhood Weight Matrices on Smoothing Properties of Conditional Autoregressive (CAR) Models." *International Journal of Health Geographics* 6 (1): 54. https://doi.org/10.1186/1476-072x-6-54.

Kassambara, Alboukadel. 2023. *Ggpubr: 'Ggplot2' Based Publication Ready Plots*. https://rpkgs.datanovia.com/ggpubr/.

Parry, Josiah, and Dexter H Locke. 2024. *Sfdep: Spatial Dependence for Simple Features*. https://sfdep.josiahparry.com.

Pebesma, Edzer. 2018. "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal* 10 (1): 439–46. https://doi.org/10.32614/RJ-2018-009.

Robinson, David, Alex Hayes, and Simon Couch. 2023. "Broom: Convert Statistical Objects into Tidy Tibbles." https://CRAN.R-project.org/package=broom.

Rowe, Francisco, and Dani Arribas-Bel. 2024. "University of Liverpool, Spatial Modeling for Data Scientists, ENVS453, Course repository." https://gdsl-ul.github.io/san/.

Stan Development Team. 2020. "RStan: The R Interface to Stan." http://mc-stan.org/.

Tobler, W. R. 1970. "A Computer Movie Simulating Urban Growth in the Detroit Region." *Economic Geography* 46 (sup1): 234–40. https://doi.org/10.2307/143141.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wood, S. N. 2011. "Fast Stable Restricted Maximum Likelihood and Marginal Likelihood Estimation of Semiparametric Generalized Linear Models" 73: 3–36. https://CRAN.R-project.org/web/packages/mgcv/index.html.

Kevin Horan
Hamilton Institute, Maynooth University
Maynooth
Co. Kildare, Ireland
https://github.com/horankev
ORCiD: 0009-0003-9378-0084
kevin.horan.2021@mumail.ie

Katarina Domijan
Department of Mathematics and Statistics, Maynooth University
Maynooth
Co. Kildare, Ireland
ORCiD: 0000-0002-4268-2236
katarina.domijan@mu.ie

Chris Brunsdon
National Centre for Geocomputation, Maynooth University
Maynooth
Co. Kildare, Ireland
ORCiD: 0000-0003-4254-1780
Christopher.Brunsdon@mu.ie