Aim of this practical:

1. Work with different types of spatial data types including: Areal, Geostatistical and Spatial Point process data.
2. Read and visualize shapefiles into R
3. Manipulate and visualize `sf` spatial object in R
4. Manipulate and visualize raster data in R.

we are going to learn:

- Explore tools for spatial data wrangling and visualization.
- Use some commonly used metrics to assess spatial autocorrelation in our data.

In this practical we will:

- Explore tools for areal spatial data wrangling and visualization.
- Compute Morans'I to identify spatial autocorrelation in our data.

0 Areal (lattice) data

Areal data our measurements are summarised across a set of discrete, non-overlapping spatial units such as postcode areas, health board or pixels on a satellite image. In consequence, the spatial domain is a countable collection of (regular or irregular) areal units at which variables are observed. Many public health studies use data aggregated over groups rather than data on individuals - often this is for privacy reasons, but it may also be for convenience.

In the next example we are going to explore data on respiratory hospitalisations for Greater Glasgow and Clyde between 2007 and 2011. The data are available from the `CARBayesdata` R Package:

```
library(CARBayesdata)

data(pollutionhealthdata)
data(GGHB.IZ)
```

The `pollutionhealthdata` contains the spatiotemporal data on respiratory hospitalisations, air pollution concentrations and socio-economic deprivation covariates for the 271 Intermediate Zones (IZ) that make up the Greater Glasgow and Clyde health board in Scotland. Data are provided by the Scottish Government and the available variables are:

- `IZ`: unique identifier for each IZ.
- `year`: the year were the measruments were taken
- `observed`: observed numbers of hospitalisations due to respiratory disease.
- `expected`: expected numbers of hospitalisations due to respiratory disease computed using indirect standardisation from Scotland-wide respiratory hospitalisation rates.
- `pm10`: Average particulate matter (less than 10 microns) concentrations.
- `jsa`: The percentage of working age people who are in receipt of Job Seekers Allowance
- `price`: Average property price (divided by 100,000).

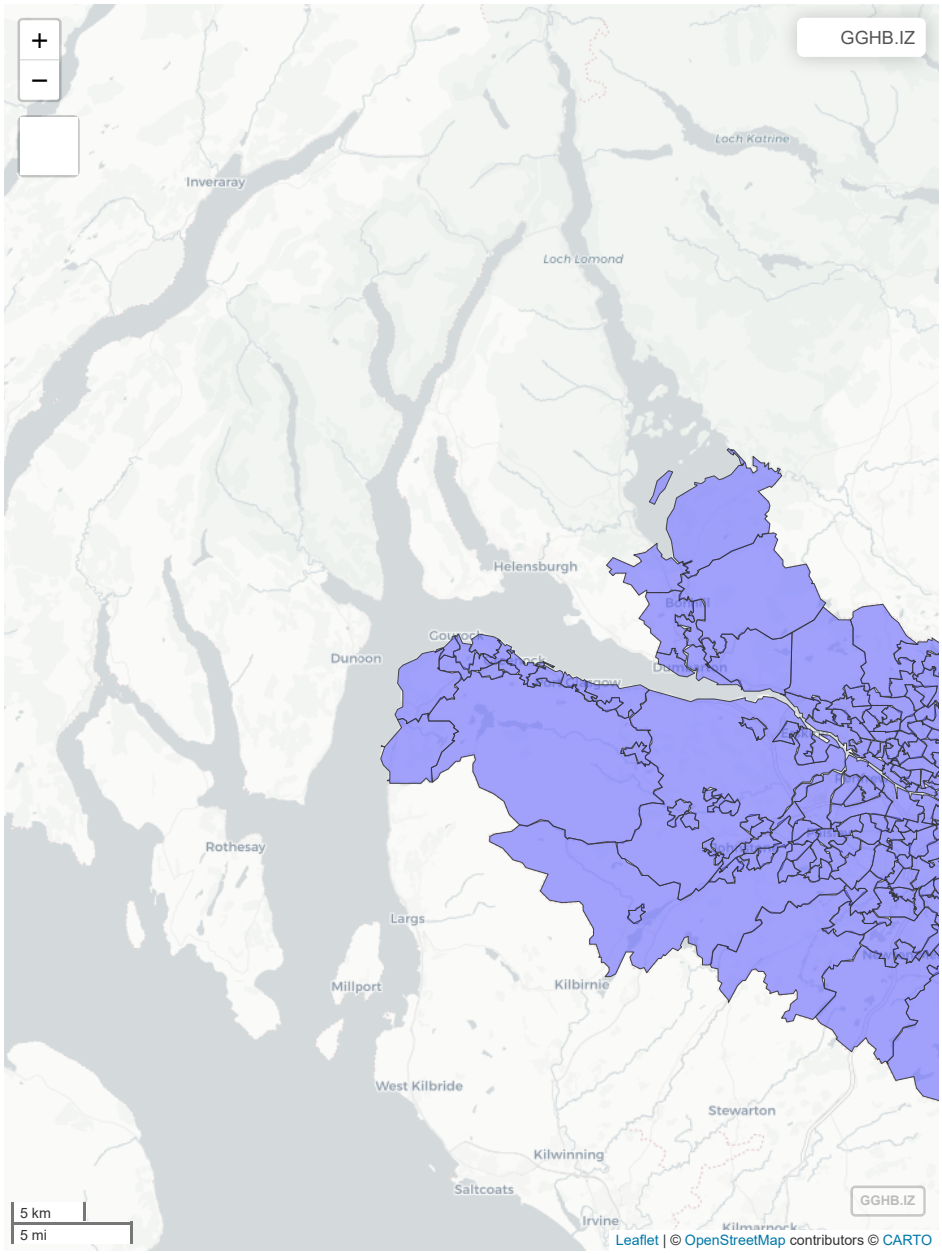


Figure 1: Greater Glasgow and Clyde health board represented by 271 Intermediate Zones

The `GGHB.IZ` data is a Simple Features (`sf`) object containing the spatial polygon information for the set of 271 Intermediate Zones (IZ), that make up of the Greater Glasgow and Clyde health board in Scotland (Figure 1).

Let's start by loading useful libraries:

```
library(sf)
library(ggplot2)
library(scico)
```

The `sf` package allows us to work with vector data which is used to represent points, lines, and polygons. It can also be used to read vector data stored as a shapefiles.

First, lets combine both data sets based on the Intermediate Zones (IZ) variable using the `merge` function from `base` R:

```
resp_cases <- merge(GGHB.IZ, pollutionhealthdata, by = "IZ")
```

In epidemiology, disease risk is usually estimated using Standardized Mortality Ratios (SMR). The SMR for a given spatial areal unit i is defined as the ratio between the observed (Y_i) and expected (E_i) number of cases:

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

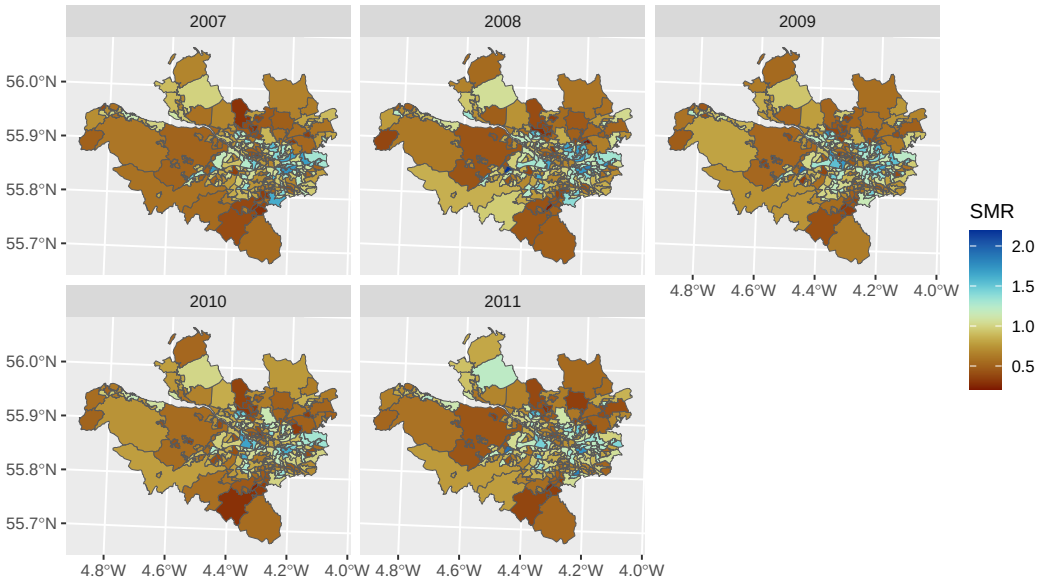
A value $SMR > 1$ indicates that there are more observed cases than expected which corresponds to a high risk area. On the other hand, if $SMR < 1$ then there are fewer observed cases than expected, suggesting a low risk area.

We can manipulate `sf` objects the same way we manipulate standard data frame objects via the `dplyr` package. Lets use the pipeline command %>% and the `mutate` function to calculate the yearly SMR values for each IZ:

```
library(dplyr)
resp_cases <- resp_cases %>%
  mutate(SMR = observed/expected, .by = year )
```

Now we use `ggplot` to visualize our data by adding a `geom_sf` layer and coloring it according to our variable of interest (i.e., SMR). We can further use `facet_wrap` to create a layer per year and chose an appropriate color palette using the `scale_fill_scico` from the `scico` package:

```
ggplot()+
  geom_sf(data=resp_cases,aes(fill=SMR))+
  facet_wrap(~year)+scale_fill_scico(palette = "roma")
```



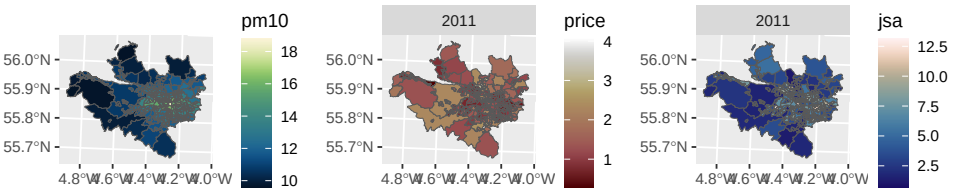
Task

Produce a map that shows the spatial distribution of each of the following variables for the year 2011:

- Average particulate matter `pm10`
- Average property price `price`
- Percentage of working age people who are in receipt of Job Seekers Allowance `jsa`

hint
You can use the `filter` function from `dplyr` to subset the data according to the year of interest.
Click here to see the solution

```
# Library for plotting multiple maps together

library(patchwork)

# subset data set for 2011

resp_cases_2011 <- resp_cases %>% filter(year ==2011)

# pm10 plot

pm10_plot <- ggplot()+
  geom_sf(data=resp_cases_2011,aes(fill=pm10))+
  scale_fill_scico(palette = "navia")

# property price

price_plot <- ggplot()+
  geom_sf(data=resp_cases_2011,aes(fill=price))+
  facet_wrap(~year)+scale_fill_scico(palette = "bilbao")

#  percentage jsa

jsa_plot <- ggplot()+
  geom_sf(data=resp_cases_2011,aes(fill=jsa))+
  facet_wrap(~year)+scale_fill_scico(palette = "lapaz")

# plot maps together

pm10_plot + price_plot + jsa_plot + plot_layout(ncol=3)
```



As with the other types of spatial modelling, our goal is to observe and explain spatial variation in our data. Generally, we are aiming to produce a smoothed map which summarises the spatial patterns we observe in our data.

A key aspect of any spatial analysis is that observations closer together in space are likely to have more in common than those further apart. This can lead us towards approaches similar to those used in time series, where we consider the spatial *closeness* of our regions in terms of a *neighbourhood structure*.

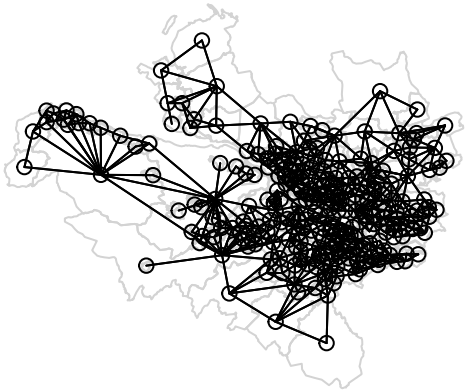
The function `poly2nb()` of the `spdep` package can be used to construct a list of neighbors based on areas with contiguous boundaries (e.g., using Queen contiguity).

```
library(spdep)

W.nb <- poly2nb(GGHB.IZ,queen = TRUE)
W.nb
```

```
Neighbour list object:
Number of regions: 271
Number of nonzero links: 1424
Percentage nonzero weights: 1.938971
Average number of links: 5.254613
2 disjoint connected subgraphs
```

The warning tell us that the neighbourhood is comprised of two interconnected regions. By looking at the neighbourhood graph below, we can see that these are the North and South Glasgow regions which are separated by the River Clyde.

```
plot(st_geometry(GGHB.IZ), border = "lightgray")
plot.nb(W.nb, st_geometry(GGHB.IZ), add = TRUE)
```



You could use the `snap` argument within `poly2nb` to set a distance at which the different regions centroids are consider neighbours. To do so we first need to be aware about the spatial units of the spatial coordinate reference system (CRS). We can check this as follows:

```
st_crs(GGHB.IZ)$units
```

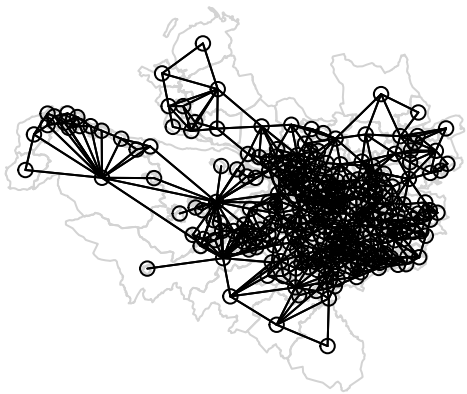
```
[1] "m"
```

Then, we could set a distance of 250m to join the IZ centroids that are are less than 250m apart.

```
W.nb250 <- poly2nb(GGHB.IZ,snap=250)
W.nb250
```

```
Neighbour list object:
Number of regions: 271
Number of nonzero links: 1758
Percentage nonzero weights: 2.393758
Average number of links: 6.487085
```

```
plot(st_geometry(GGHB.IZ), border = "lightgray")
plot.nb(W.nb250, st_geometry(GGHB.IZ), add = TRUE)
```



Once we have identified a set of neighbours using our chosen method, we can use this to account for correlation.

Moran's I is a measure of global spatial autocorrelation, and can be considered an extension of the Pearson correlation coefficient. For a set of data $Z_1, \dots, Z_m$ measured on regions $B_1, \dots B_m$, with neighbourhood matrix W, we can compute Moran's I as:

$$I = \frac{m}{\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}} \frac{\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}(Z_i - \bar{Z})(Z_j - \bar{Z})}{\sum_{i=1}^{m}(Z_i - \bar{Z})^2}$$

This is basically a function of differences in values between neighbouring areas. By far the most common approach is to use a binary neighbourhood matrix, W, denoted by

$$w_{ij} = \begin{cases} 1 & \text{if areas } (B_i, B_j) \text{ are neighbours.} \\ 0 & \text{otherwise.} \end{cases}$$

Binary matrices are used for their simplicity. Fitting spatial models often requires us to invert W, and this is less computationally intensive for sparse matrices.

Moran's I ranges between -1 and 1, and can be interpreted in a similar way to a standard correlation coefficient.

- $I = 1$ implies that we have **perfect spatial correlation**.
- $I = 0$ implies that we have **complete spatial randomness**.
- $I = -1$ implies that we have **perfect dispersion** (negative correlation).

Our observed I is a point estimate, and we may also wish to assess whether it is significantly different from zero. We can test for a statistically significant spatial correlation using a permutation test, with hypotheses:

$$\begin{aligned} H_0 &: \text{ negative or no spatial association } (I \leq 0) \\ H_1 &: \text{ positive spatial association } (I > 0) \end{aligned}$$

We can use `moran.test()` to test this hypothesis by setting `alternative = "greater"`. To do so, we need to supply list containing the neighbors via the `nb2listw()` function from the `spdep` package. Lets assess now the spatial autocorrelation of the SMR in 2011:

```
# subset the data
resp_cases_2011 <- resp_cases %>% filter(year ==2011)

# neighbors list
nbw <- nb2listw(W.nb, style = "W")

# Global Moran's I
gmoran <- moran.test(resp_cases_2011$SMR, nbw,
                     alternative = "greater")
gmoran
```

```
    Moran I test under randomisation

data:  resp_cases_2011$SMR
weights: nbw

Moran I statistic standard deviate = 11.42, p-value < 2.2e-16
```

```
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.439899780      -0.003703704       0.001508809
```

> Question
>
> What do we conclude from the Moran's I test?
> Answer
> Since have set the alternative hypothesis to be $ I > 0 $ and have a *p*-value < 0.05, we then reject the null hypothesis and conclude there is evidence for positive spatial autocorrelation.

A local version of Moran's I can also be used to measure the similarity between each IZ via the `localmoran` function:

```
lmoran <- localmoran(resp_cases_2011$SMR, nbw, alternative = "two.sided")
```

In this case we set `alternative = "two.sided"` to test whether there is any evidence of spatial autocorrelation in our data:

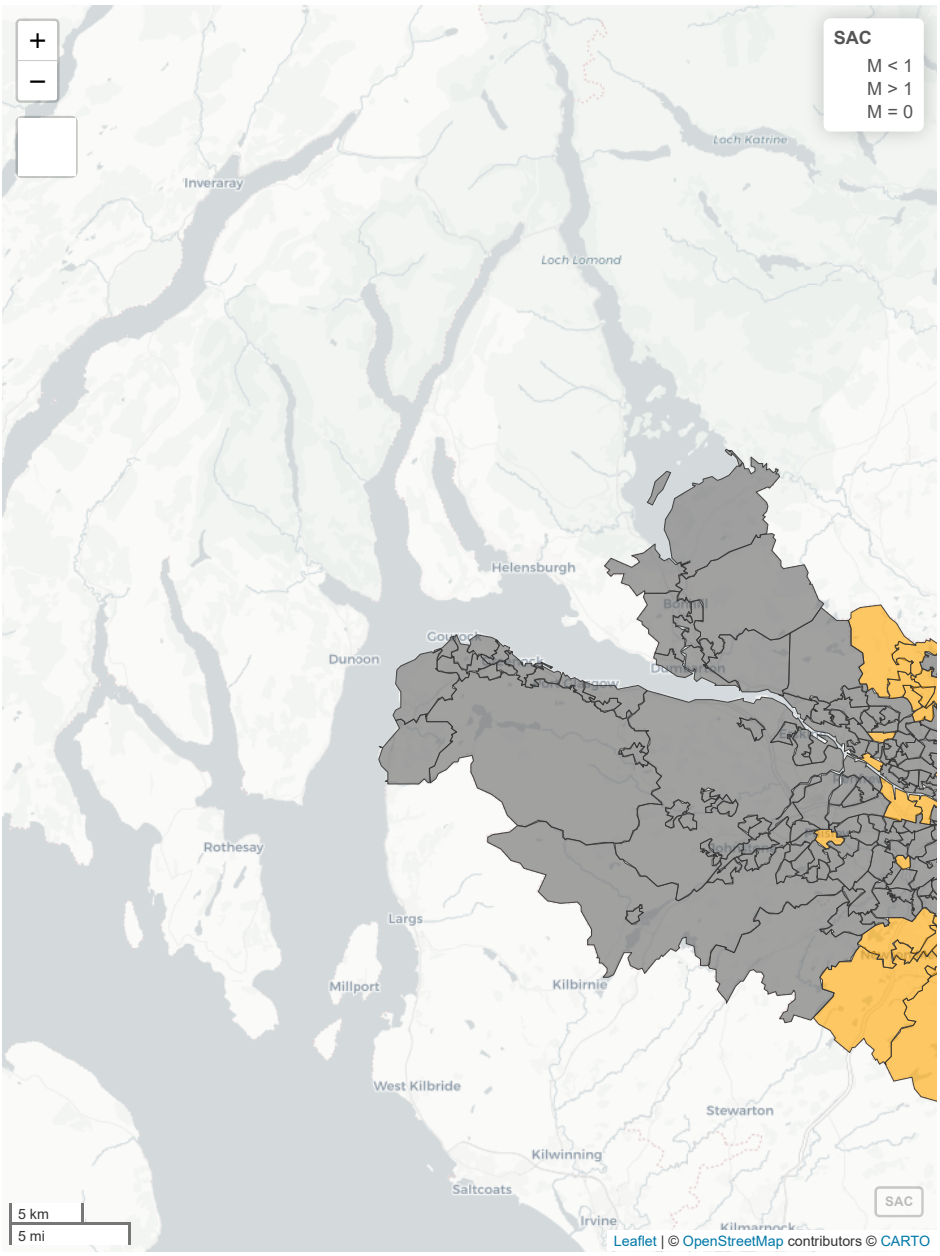
$$H_0: \text{ no spatial association } (I = 0)$$
$$H_1: \text{ some spatial association } (I \neq 0)$$

We can obtain the Z-scores from the test (`Z.Ii`) where any values smaller than –1.96 indicate negative spatial autocorrelation, and z-score values greater than 1.96 indicate positive spatial autocorrelation:

```
resp_cases_2011_m <- resp_cases_2011 %>% mutate(Zscores = lmoran[,"Z.Ii"])

resp_cases_2011_m <- resp_cases_2011_m %>%
  mutate (SAC = case_when(
  Zscores > qnorm(0.975) ~ " M > 1" ,
  Zscores < -1*qnorm(0.975) ~ " M < 1",
  .default = "M = 0"),
  SAC=as.factor(SAC)
)
```

We can visualize these results using either `ggplot` as we did before, or via the `mapview` library which contains interactive features:

```
library(mapview)
mapview(resp_cases_2011_m,
        zcol = "SAC",
        layer.name = "SAC",
        col.regions=c("turquoise","orange","grey40"))
```



In this practical we will:

- Explore tools for geostatistical spatial data wrangling and visualization.
- Compute a variogram to assess for spatial autocorrelation in our data.

First, lets load some useful libraries for data wrangling and visualization

```
# For plotting
library(mapview)
library(ggplot2)
library(scico) # for colouring palettes

# Data manipulation
library(dplyr)
```

0 Geostatistical data

Tobler's first law of geography states that:

"*Everything is related to everything else, but near things are more related than distant things*"

Spatial patterns are fundamental in environmental and ecological data. In many ecological and environmental settings, measurements from fixed sampling units, aiming to quantify spatial variation and interpolate values at unobserved sites.

Geostatistical data are the most common form of spatial data found in environmental setting. In these data we regularly take measurements of a spatial ecological or environmental process at a set of fixed locations. This could be data from transects (e.g, where the height of trees is recorded), samples taken across a region (e.g., water depth in a lake) or from monitoring stations as part of a network (e.g., air pollution). In each of these cases, our goal is to estimate the value of our variable across the entire space.

Let D be our two-dimensional region of interest. In principle, there are infinite locations within D, each of which can be represented by mathematical coordinates (e.g., latitude and longitude). We then can identify any individual location as $s_i = (x_i, y_i)$, where x_i and y_i are their coordinates.

We can treat our variable of interest as a random variable, Z which can be observed at any location as $Z(\mathbf{s}_i)$.

Our geostatistical process can therefore be written as:

$$\{Z(\mathbf{s}); \mathbf{s} \in D\}$$

In practice, our data are observed at a finite number of locations, m, and can be denoted as:

$$z = \{z(\mathbf{s}_1), \dots z(\mathbf{s}_m)\}$$

In the next example, we will explore data on the Pacific Cod (*Gadus macrocephalus*) from a trawl survey in Queen Charlotte Sound. The `pcod` dataset is available from the `sdmTMB` package and contains the presence/absence records of the Pacific Cod during each surveys along with the biomass density of Pacific cod in the area swept (kg/Km^2). The `qcs_grid` data contain the depth values stored as 2×2 km grid for Queen Charlotte Sound.

```
library(sdmTMB)

pcod_df = sdmTMB::pcod
qcs_grid = sdmTMB::qcs_grid
```

0.2.1 Georeferrenced data

Let's create an initial `sf` spatial object using the standard geographic coordinate system (`EPSG:4326`). This correctly defines the point locations based on latitude and longitude.

```
library(sf)
pcod_sf =   st_as_sf(pcod_df, coords = c("lon","lat"), crs = 4326)
```

Now we can transform to the standard UTM Zone 9N projection (EPSG:32609) which uses meters:

```
pcod_sf_proj <- st_transform(pcod_sf, crs = 32609)
st_crs(pcod_sf_proj)$units
```

```
[1] "m"
```

We can change the spatial units to *km* to better reflect the scale of our ecological study and to make resulting distance/area values more intuitive to interpret:

```
pcod_sf_proj = st_transform(pcod_sf_proj,
                            gsub("units=m","units=km",
                                 st_crs(pcod_sf_proj)$proj4string))
st_crs(pcod_sf_proj)$units
```

```
[1] "km"
```

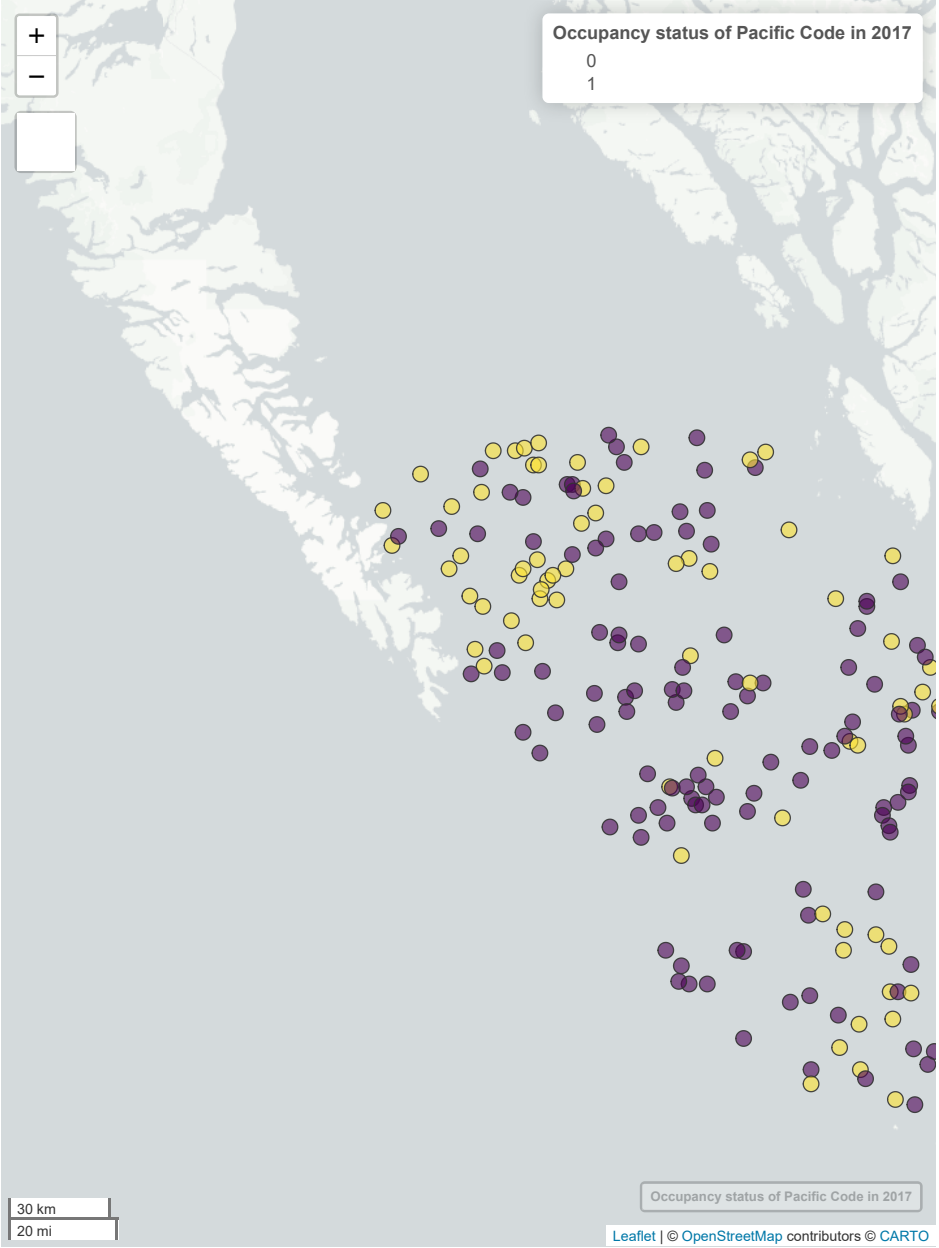
Instead of first setting an EPSG code and then transforming, we can define the target Coordinate Reference System (CRS) directly using a `proj4string`. This allows us to customize non-standard parameters in a single step, in this case, explicitly setting the projection units to kilometers (`+units=km`).

```
pcod_sf = st_transform(pcod_sf,
                       crs = "+proj=utm +zone=9 +datum=WGS84 +no_defs +type=crs +units=km" )
st_crs(pcod_sf)$units
```

```
[1] "km"
```

Spatial `sf` objects can be manipulated the same way we manipulate standard data frame objects via the `dplyr` package. For example, you can select a specific year using the `filter` function from `dplyr`. Let's map the present/absence of the Pacific Cod in 2017 using the `mapview` function:

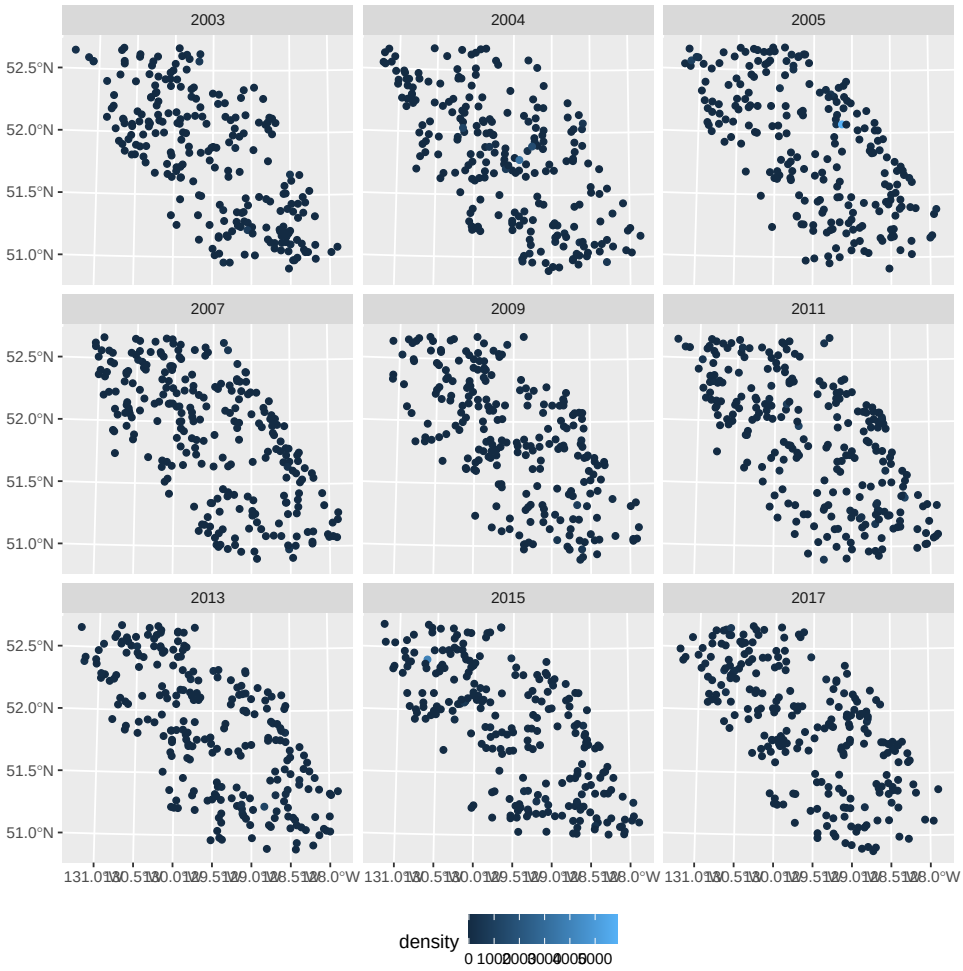
```
pcod_sf %>%
  filter(year== 2017) %>%
  mutate(present = as.factor(present)) %>%
mapview(zcol = "present",
        layer.name = "Occupancy status of Pacific Code in 2017")
```



Task

Use `ggplot` and the `sf` library to map the biomass density of the pacific code across years.
hint
You can plot an`sf` object by adding a `geom_sf` layer to a ggplot object. You can also use the `facet_wrap` argument to plot an arrange of plots according to a grouping variable.
Click here to see the solution

```
ggplot()+
  geom_sf(data=pcod_sf,aes(color=density))+
  facet_wrap(~year)+
  theme(legend.position = "bottom")
```



0.2.2 Raster Data

Environmental data is typically stored in raster format, which represents spatially continuous phenomena by dividing a region into a grid of equally-sized cells, each storing a value for the variable of interest. In R, the `terra` package is a modern and powerful tool for efficiently working with raster data. The function `rast()`, can be used both to read raster files from standard formats (e.g., `.tif` or `.tiff`) and to create a new raster object from a data frame. For instance, the following code creates a raster from the `qcs_grid` grid data for Queen Charlotte Sound.

```
library(terra)
depth_r <- rast(qcs_grid, type = "xyz")
depth_r
```

```
class       : SpatRaster
size        : 102, 121, 3  (nrow, ncol, nlyr)
resolution  : 2, 2  (x, y)
```

```
extent      : 341, 583, 5635, 5839  (xmin, xmax, ymin, ymax)
coord. ref. :
source(s)   : memory
names       :    depth, depth_scaled, depth_scaled2
min values  :  12.0120,    -6.000040,  4.892624e-08
max values  : 805.7514,     3.453937,  3.600048e+01
```

The raster object contains three layers corresponding to the (i) depth values, (ii) the scaled depth values and (iii) the squared depth values.

Notice that there are no CRS associated with the raster. Thus, we can assign appropriate CRS using the `crs` function. Additionally, we also want the raster CRS to match the CRS in the survey data (recall that we have previously reprojected our data to utm coordinates). We can assign an appropiate CRS that matches the CRS of the `sf` object as follows:

```
crs(depth_r) <- crs(pcod_sf)
```

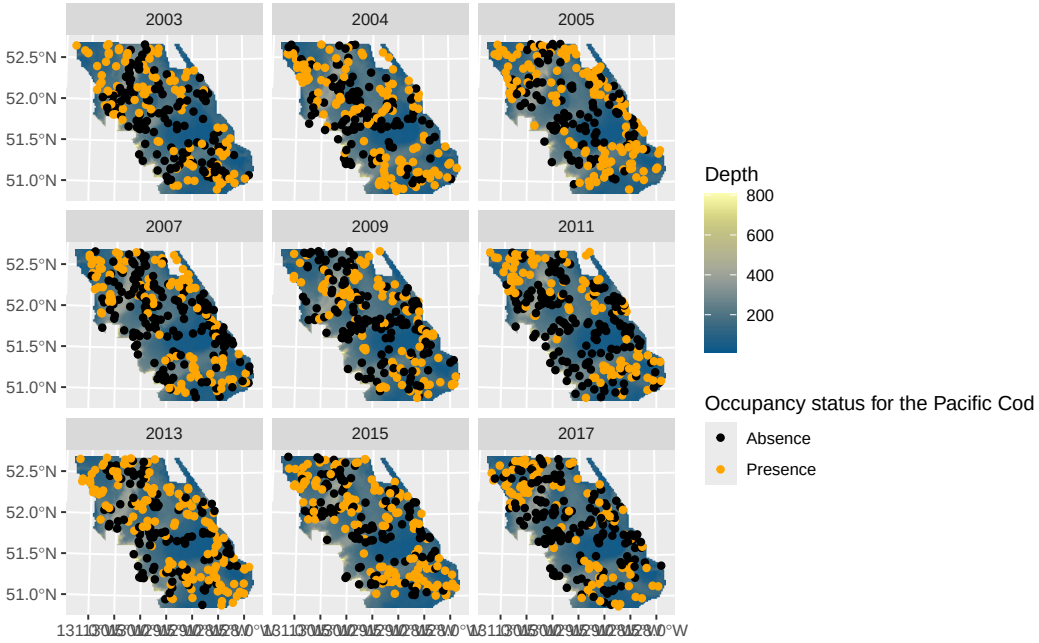
We can use the `tidyterra` package to plot raster data using `ggplot` by adding a `geom_spatraster` function and then select an appropriate `fill` and `color` palettes:

```
library(tidyterra)
```

```
Attaching package: 'tidyterra'
```

```
The following object is masked from 'package:stats':

    filter
```

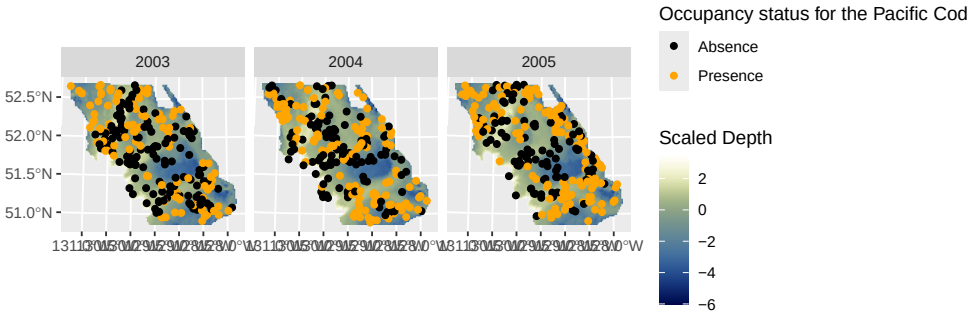
```
ggplot()+
  geom_spatraster(data=depth_r$depth)+
  geom_sf(data=pcod_sf,aes(color=factor(present))) +
  facet_wrap(~year)+
    scale_color_manual(name="Occupancy status for the Pacific Cod",
                       values = c("black","orange"),
                       labels= c("Absence","Presence"))+
  scale_fill_scico(name = "Depth",
                   palette = "nuuk",
                   na.value = "transparent" )
```



Task

Map the scaled depth and the presence/absence records of the Pacific cod for 2003 to 2005 only.
hint
The different layers of a raster can be accessed using the $ symbol.
Click here to see the solution

```
ggplot()+
  geom_spatraster(data=depth_r$depth_scaled)+
  geom_sf(data=pcod_sf %>% filter(year %in% 2003:2005),
          aes(color=factor(present)))+
  facet_wrap(~year)+
  scale_color_manual(name="Occupancy status for the Pacific Cod",
                     values = c("black","orange"),
                     labels= c("Absence","Presence"))+
    scale_fill_scico(name = "Scaled Depth",
                     palette = "davos",
                     na.value = "transparent" )
```



0.2.3 Autocorrelation and Variograms

Spatial statistics quantifies the fundamental principle that nearby things are more related. This spatial dependence means that data points are not independent, as assumed by most classical statistical models, but are instead correlated based on their proximity. While this correlation can be a valuable source of information, it must be explicitly accounted for to avoid biased inference and incorrect conclusions.

The first step is to assess whether there is any evidence of spatial dependency in our data. Spatial dependence can be explored by a function known as a variogram $2\gamma(\cdot)$ (or semivariogram $\gamma(\cdot)$). The variogram is similar in many ways to the autocorrelation function used in time series modelling. In simple terms, it is a function which measures the difference in the spatial process between a pair of locations a fixed distance apart

The variogram measures the variance of the difference in the process $Z(\cdot)$ at two spatial locations $\mathbf{s}$ and $\mathbf{s}+\mathbf{h}$ and is defined as :

$$\text{Var}[Z(\mathbf{s}) - Z(\mathbf{s}+\mathbf{h})] = E[(Z(\mathbf{s}) - Z(\mathbf{s}+\mathbf{h}))^2] = 2\gamma_z(\mathbf{h}).$$

Here, $2\gamma_z(\mathbf{h})$ is the variogram, but in practice we use the semi-variogram, $\gamma_z(\mathbf{h})$. We use the semi-variogram because our points come in pairs, and the semi-variance is equivalent to the variance per point at a given lag.

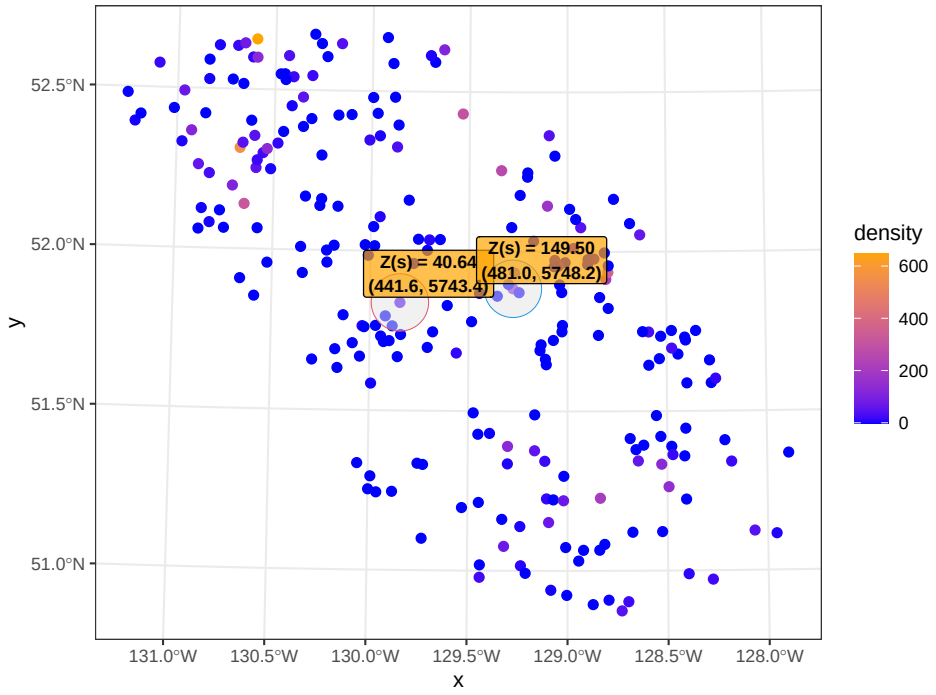
- When the variance of the difference $Z(\mathbf{s}) - Z(\mathbf{t})$ is relatively small, then $Z(\mathbf{s})$ and $Z(\mathbf{t})$ are similar (spatially correlated).
- When the variance of the difference $Z(\mathbf{s}) - Z(\mathbf{t})$ is relatively large, then $Z(\mathbf{s})$ and $Z(\mathbf{t})$ are less similar (closer to independence).

The variogram is a function of the underlying geostatistical process Z. In practice, we only have access to m realisations of this process, and therefore we have to estimate the variogram. This is known as the empirical variogram.

We obtain this by computing the semi-variance for all possible pairs of observations: $\gamma(\mathbf{s},\mathbf{t}) = 0.5(Z(\mathbf{s}) - Z(\mathbf{t}))^2$.

Example

To illustrate how an empirical variogram is computed, consider the biomass density of the Pacific Cod in 2017 for the two highlighted locations below.



1. We can first compute the distance between the two locations using the standard Euclidean distance formula as

$$h = \sqrt{(441.6 - 481)^2 + (5743.4 - 5748.2)^2} \approx 39 \text{ Km}$$

2. Next, we compute the semi-variance between the points using their observed values as

$$\gamma(\mathbf{s}, \mathbf{t}) = 0.5(Z(\mathbf{s}) - Z(\mathbf{t}))^2 = 0.5(149.5 - 40.64)^2 = 5925.25$$

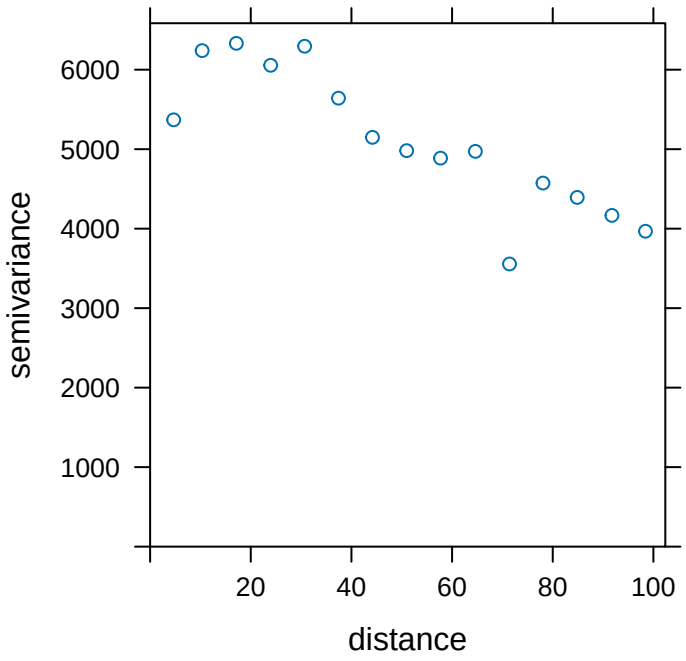
3. We repeat this process for every possible pair of points, and plot h against $\gamma(\mathbf{s}, \mathbf{t})$ for each.

To make the variogram easier to use and interpret, we divide the distances into a set of discrete bins, and compute the average semi-variance in each. We compute this binned empirical variogram as:

$$\gamma(\mathbf{h}) = \frac{1}{2N(h_k)} \sum_{(\mathbf{s},\mathbf{t}) \in N(h_k)} [z(\mathbf{s}) - z(\mathbf{t})]^2$$

We can calculate the binned- empirical variogram for the data using `variogram` function from the `gstat` library. This plot shows the semi-variances for each pair of points. Lets compute a variogram for the biomass density of the Pacific Cod in 2017:

```r
library(gstat)

pcod_sf_subset <- pcod_sf %>% filter(year ==2017)

vgm1 <- variogram(density~1, pcod_sf_subset)
plot(vgm1)
```

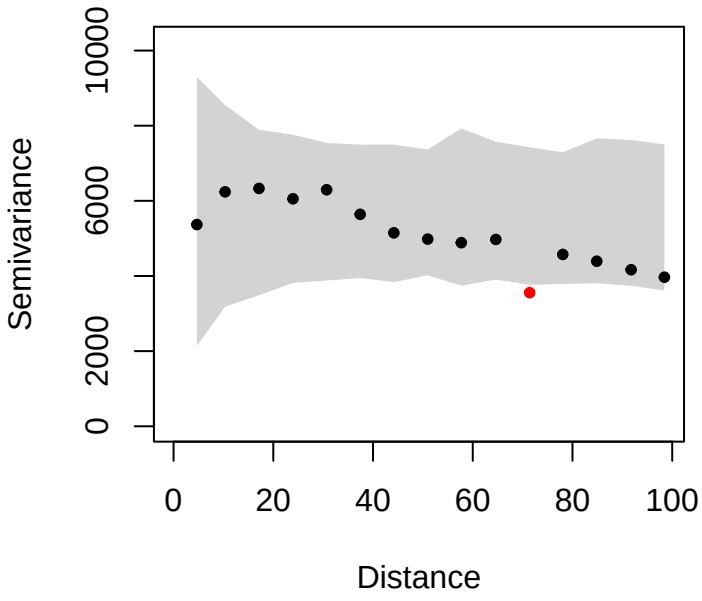


Assessing spatial dependence

We can construct null envelope based on permutations of the data values across the locations, i.e. envelopes built under the assumption of no spatial correlation. By overlapping these envelopes with the empirical variograms we can determine whether there is some spatial dependence in our data ,e.g. if our observed variograms falls outside of the envelopes constructed under spatial randomness.

We can construct permutation envelopes on the gstat empirical variogram using the `envelope` function from the `variosig` R package. Then we can visualize the results using the `envplot` function:

```r
library(variosig)

varioEnv <- envelope(vgm1,
                     data = pcod_sf_subset,
                     locations = st_coordinates(pcod_sf_subset),
                     formula = density ~ 1,
                     nsim = 499)

envplot(varioEnv)
```




```
[1] "There are 1 out of 15 variogram estimates outside the 95% envelope."
```

In this practical we will:

- Explore tools for spatial point pattern data wrangling and visualization.

First, lets load some useful libraries:

```
# For plotting
library(ggplot2)
library(scico) # for colouring palettes

# Data manipulation
library(dplyr)
```

0 Spatial Point processes data

In point processes we measure the locations where events occur and the coordinates of such occurrences are our data.

Point process models are probabilistic models that describe the likelihood of *patterns* of points that represent the random location of some event. A spatial point process is a set of locations that have been generated by some form of stochastic (random) mechanism. In other words, the point process is a random variable operating in continuous space, and we observe realisations of this variable as point patterns across space (and/or time).

Consider a fixed geographical region A. The set of locations at which events occur are denoted $\mathbf{s} = s_1, \dots, s_n$. We let $N(A)$ be the random variable which represents the number of events in region A.

Our primary interest is in measuring where events occur, so the locations are our data. We typically assume that a spatial point pattern is generated by an unique point process over the whole study area. This means that the delimitation of the study area will affect the observed point patters.

The observed distribution of points can be described based on the intensity of points within a delimited region. We can define the (first order) intensity of a point process as the expected number of events per unit area. This can also be thought of as a measure of the density of our points. In some cases, the intensity will be constant over space (homogeneous), while in other cases it can vary by location (inhomogeneous or heterogenous).

In the next example, we will explore tools for visualizing spatial point patterns. Specifically, we will map the spatial distribution of the Ringlet butterfly in Scotland's Cairngorms National Park (CNP).

0.3.1 BNM citizen science program

Citizen science initiatives have become an important source of information in ecological research, offering large volumes of species distribution data collected by volunters for multiple taxonomic groups across wide spatial and temporal scales

Butterflies for the New Millennium (BNM) is a large-scale monitoring scheme launched in the earlies 70's to keep track of butterflies' populations in the UK. With over 12 million butterflies sighting and more than 10,000 volunteers, this recording scheme has proven to be a successful program that has been used to assess long-term changes in the distributions of UK butterfly species.

Here we will focus on the distribution of the Ringlet butterfly species, which holds particular significance in environmental studies as one of the *Habitat specialists* species (UK Government, 2024). The data set consists of Ringlet butterfly presence-only records collected by volunteers in Scotland's Cairngorms National Park (CNP).

Reading shapefiles into R

First, we load the geographical region of interest which can be downloaded here (i.e., CNP boundaries). We can use thre `st_read` function from the `sf` library to load the `.shp` file by specifying the directory where you downloaded the files:

```
library(sf)
shp_SGC <-  st_read("datasets/SG_CairngormsNationalPark/SG_CairngormsNationalPark_2010.shp",
```

Then, we can use appropriate CRS for the UK (i.e., EPSG code: 27700) :

```
shp_SGC <- shp_SGC %>% st_transform(crs = 27700)
st_crs(shp_SGC)$units
```

```
[1] "m"
```

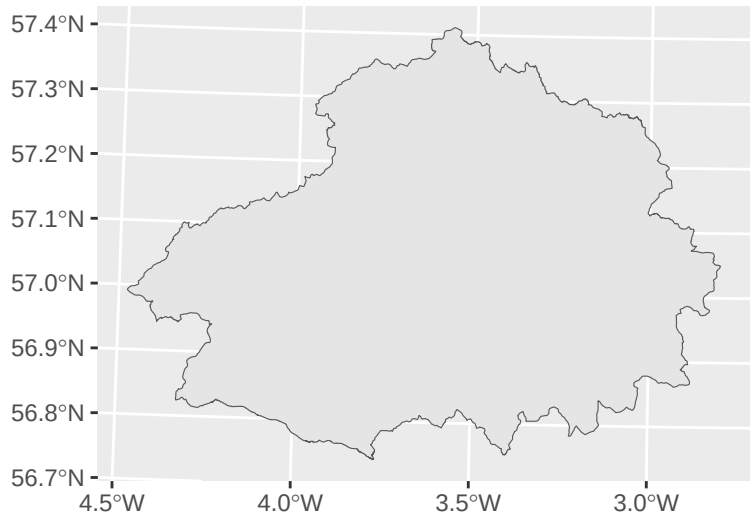
Notice that the spatial resolution is in meters. Let's change the spatial units to *km* to make resulting distance/area values more intuitive to interpret:

```
shp_SGC <- st_transform(shp_SGC,gsub("units=m","units=km",st_crs(shp_SGC)$proj4string))
st_crs(shp_SGC)$units
```

```
[1] "km"
```

We can then plot the CNP boundary as follows:

```
ggplot()+
  geom_sf(data=shp_SGC)
```



Creating sf spatial objects in R

Now we will read the Ringlet butterfly records which can be downloaded below:

```
ringlett <- read.csv("datasets/bnm_ringlett.csv")
head(ringlett)
```

```
         y         x
1 57.58752 -2.712498
2 54.97742 -3.274879
3 54.89929 -3.771451
4 55.40323 -5.737059
5 54.91438 -3.959336
6 55.87255 -4.167174
```

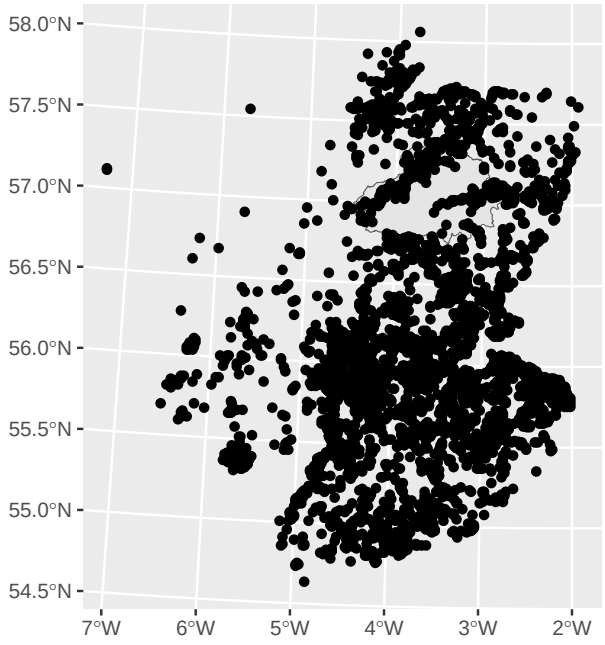
The data set contains the longitude latitude where an observation was made. We can convert this into a spatial `sf` object using the `st_as_sf` function by declaring the columns in our data that contain the spatial coordinates:

```
ringlett_sf <- ringlett %>% st_as_sf(coords = c("x","y"),crs = "+proj=longlat +datum=WGS84")
```

Task

We have set standard WGS84 coordinates for the Ringlet butterfly occurrence records. Set the CRS to match the CRS used in shapefile. Then, produce a map of the CNP region with the projected observations overlayed.
Take hint
You can use the `st_transform()` function to change the coordinates of an `sf` object (type `?st_transform` for more details)/
Click here to see the solution

```
ringlett_sf <- ringlett_sf %>%
  st_transform(st_crs(shp_SGC))

ggplot()+
  geom_sf(data=shp_SGC)+
  geom_sf(data=ringlett_sf)
```

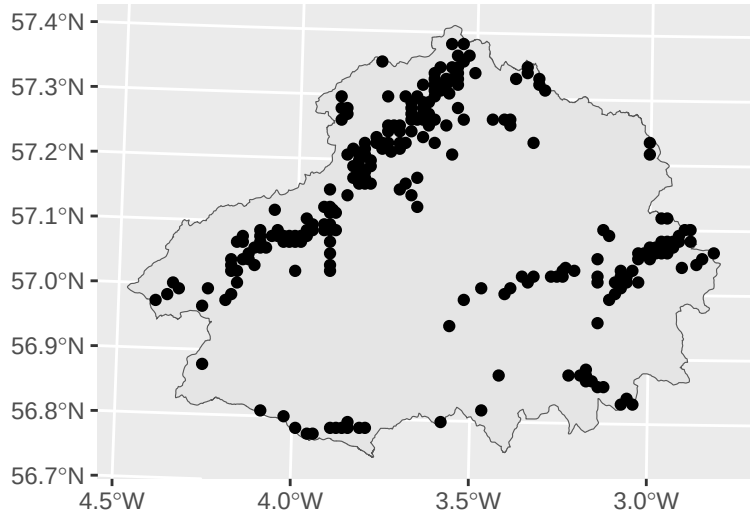


We can subset two `sf` objects with the **same** CRS in the same way as we subset a data frame in R. For example, if we want to subset the Ringlet butterfly occurrence records to those contained only within the CNP, we can type the following:

```
ringlett_CNP <- ringlett_sf[shp_SGC,] # crop to mainland
```

If we plot the `ringlett_CNP` object along with the CNP boundary, we should then obtain a map of the occurrence records within the park:

```
ggplot()+
  geom_sf(data=shp_SGC)+
  geom_sf(data=ringlett_CNP)
```

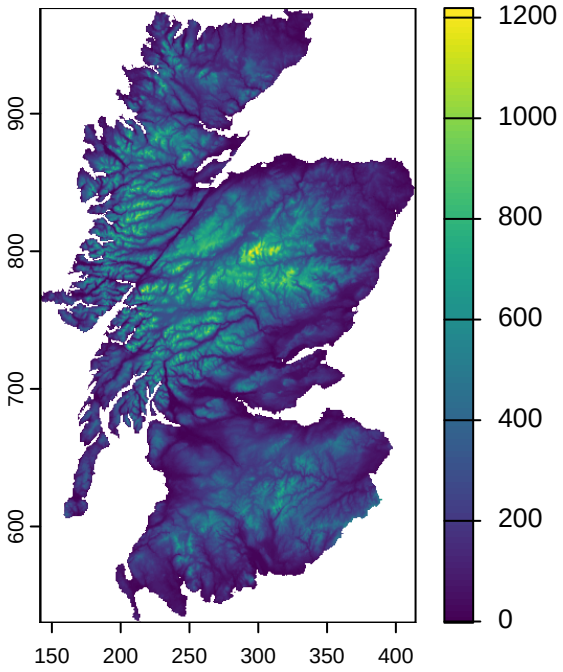


Reading Raster Data

We can use the `terra` R package to read raster files. The `Scotland_elev.tiff` raster contains the output of a digital elevation model for Scotland:

Once you download the raster you can read it using the `rast` function after specifying the path where the file has been stored. Then, we assign the same CRS as our data.

```
library(terra)
elevation_r <- rast("datasets/Scotland_elev.tiff")
crs(elevation_r) = crs(shp_SGC)
plot(elevation_r)
```

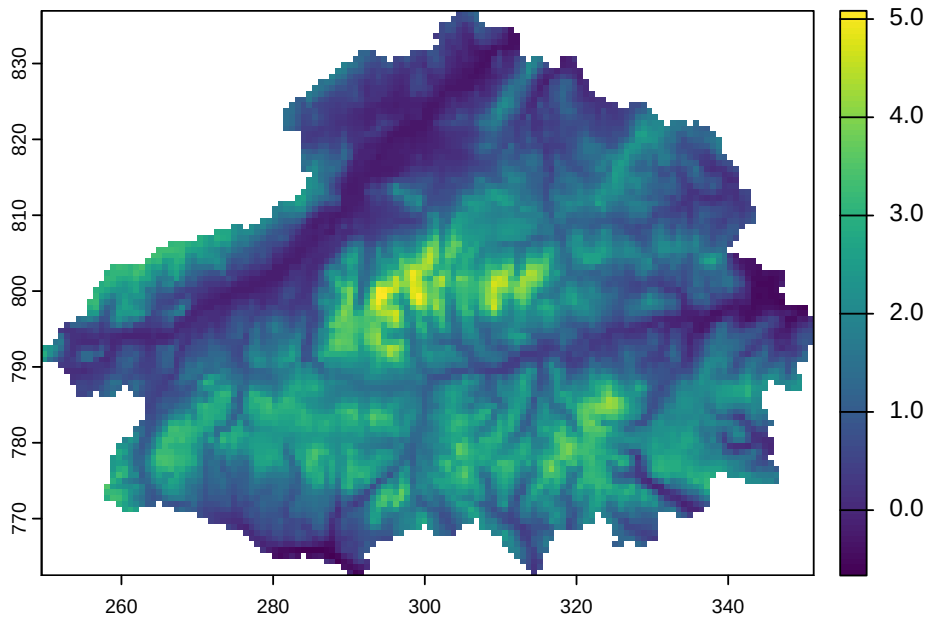


We can apply different R functions to our rasters. For example, we can scale the elevation values as follows:

```
elevation_r <- elevation_r %>% scale()
```

Lastly, we can crop the raster to the boundaries of our region of interest. Let's crop the elevation raster to the CNP area using the `crop` function:

```
elev_CNP <- terra::crop(elevation_r,shp_SGC,mask=T)
plot(elev_CNP)
```



Task

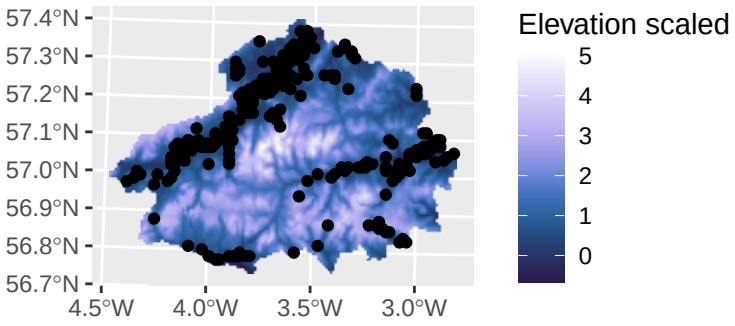
Using `tidyterra` and `ggplot`, produce a map of the elevation profile in the CNP and overlay the spatial point pattern of the Ringlet butterfly occurrence records. Use an appropriate colouring scheme for the elevation values. Do you see any pattern?

Take hint

You can use the `geom_spatraster()` to add a raster layer to a ggplot object. Furthermore the `scico` library contains a nice range of coloring palettes you can choose, type `scico_palette_show()` to see the color palettes that are available.

Click here to see the solution

```
library(scico)

ggplot()+
  tidyterra::geom_spatraster(data=elev_CNP)+
  geom_sf(data=ringlett_CNP)+
  scale_fill_scico(name = "Elevation scaled",
                   palette = "devon",
                   na.value = "transparent" )
```


57.4°N
57.3°N
57.2°N
57.1°N
57.0°N
56.9°N
56.8°N
56.7°N
4.5°W
4.0°W
3.5°W
3.0°W
Elevation scaled
5
4
3
2
1
0