vignetteDownscaler

```r
source("fnCheckInputsDown.R")
source("fnCreateMesh.R")
source("fnPredictDown.R")
```

Specifying the arguments of `fnPredictDown()`

`fnPredictDown()` aims to predict spatial data when areal data and point data within the observed areal data available. The prediction is made by the following model:

$$Y(s_i) = \beta_0(s_i) + \beta_1(s_i)X(s_i) + \epsilon_i$$

where $Y(s_i)$ is the point data and $X(s_i)$ is the observed areal data that the point $Y(s_i)$ locates.

Thus, To use `fnPredictDown()`, the observed point (`depoint`) and areal data (`dearea`) are compulsory. They are `sf` objects.

Then, we need to specify the locations or areas where we want to get predictions. Depending on where we want to predict,we would need to specify `sf` objects`dppoint` for a set of points and `dparea` for a set of areas.

`dppoint` can be an `sf` object containing a few points or a dense set of regular points within the study region representing a continuous surface (`dpcontsurface`).

We also need to specify an `sf` object the boundary of the region of study, `boundaryregion`.

Note that all objects need to be `sf` objects and have the same projection. Therefore, if we have `raster` or `sp` data, we need to transform them to `sf` to be able to use the package.

Specifying the observed data

Similar to the vignette of `fnPredictMelding()`, we will show how to predict PM 2.5 in UK. The projection method is 4326 or WSG84 (latitude and longitude).

We obtain the boundary of the region of study with the `rgeoboundaries` package and transform it to the chosen projection. Then we construct `sf` objects from the observed point data and areal data. Please read the previous vignette (Link), if you do not know how to construct `sf` objects and specify the estimated data.

The column names of `depoint` data have to be `value`,`geometry`, where `value` stands for the estimated value for point data and the other column includes spatial information.

The column names of `dearea` data have to be `value`,`geometry`, where `value` stands for the estimated value for areal data and the other column includes spatial information.

```r
# CRS projection
crsproj <- 4326

library(rgeoboundaries)
boundaryregion <- geoboundaries("United Kingdom")
boundaryregion <- st_transform(boundaryregion , crsproj)[,6]
```

```r
# observed areal data
dearea <- st_read("areal_data/dearea.shp")
```

```
## Reading layer `dearea' from data source `C:\SpatialM\areal_data\dearea.shp' using driver `ESRI Shape
## Simple feature collection with 92 features and 1 field
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: -8.05 ymin: 50.05 xmax: 1.55 ymax: 60.85
## geographic CRS: GCS_WGS_84_with_axis_order_normalized_for_visualization
```

```r
# observed point data
depoint <- read.csv("pointdata.csv")
depoint <- depoint[, c(2, 3, 4)]
colnames(depoint) <- c('value', 'y', 'x')

depoint <- depoint %>% st_as_sf(coords = c("x", "y"), dim = "XY") %>%
  st_set_crs(crsproj) %>% st_cast("MULTIPOINT")

colnames(dearea) <- c('value', 'geometry')
```

Specifying the point data and areal data for prediction

Areal data for prediction `dparea` is an `sf` object with columns `value` and `geometry`. The estimated point position and area will be the predicted point and area.

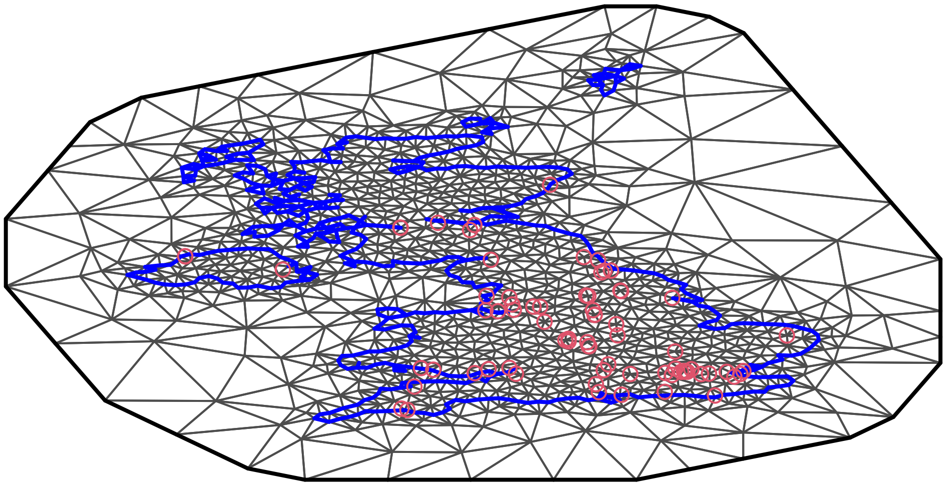
```r
dppoint <- depoint
dparea <- dearea
```

If we wish to predict in a continuous surface, we need to provide a dense grid of points within the region of study. The method can be found (Link).

Downscaler Method with `fnPredictDown()`

```r
# Create mesh
mesh <- fnCreateMesh(depoint, boundaryregion)
plot(mesh)
points(as.matrix(st_coordinates(depoint)[ , c(1, 2)]), col = 2)
```

Constrained refined Delaunay triangulation



```
respre <- fnPredictDown(depoint = depoint, dearea = dearea, dppoint = dppoint, dparea = dparea, boundary
```

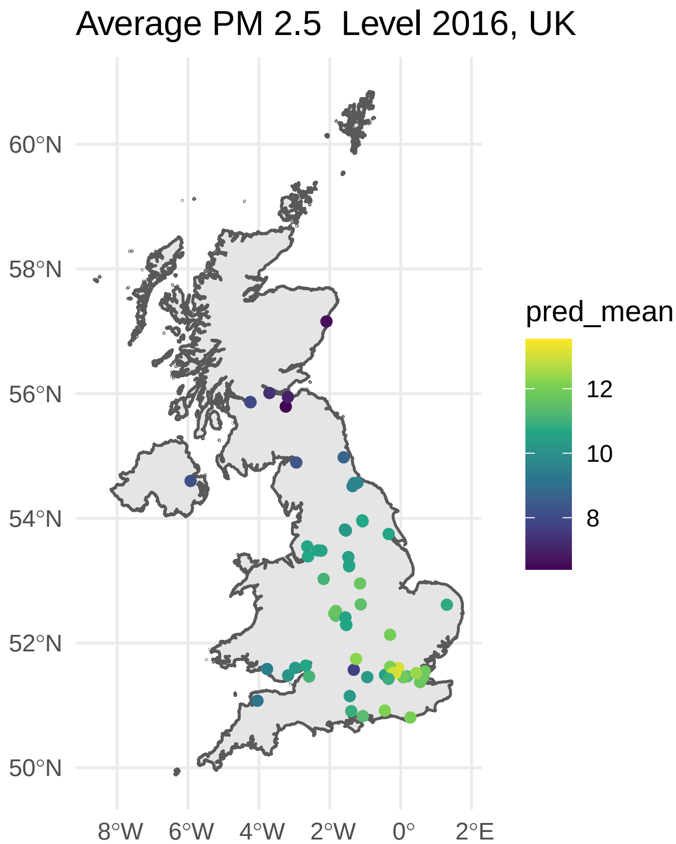
```
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
## although coordinates are longitude/latitude, st_intersects assumes that they are planar
```

```
head(respre)
```

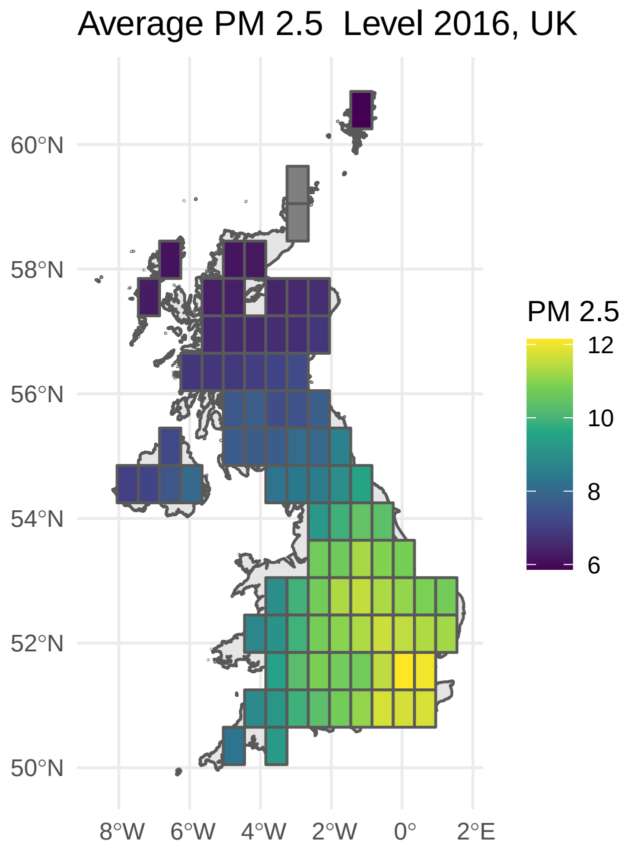
```
## [[1]]
## Simple feature collection with 64 features and 7 fields
## geometry type:  MULTIPOINT
## dimension:      XY
## bbox:           xmin: -5.928833 ymin: 50.80578 xmax: 1.301976 ymax: 57.15736
## geographic CRS: WGS 84
## First 10 features:
##      value.x   pvalue   value.y   avalue                      geometry
## 1   5.371804 5.371804  3.479238  3.479238 MULTIPOINT ((-2.094278 57.1...
## 2   2.549784 2.549784  4.772653  4.772653 MULTIPOINT ((-3.2429 55.792...
```

```
## 3  10.701091 10.701091  5.851172  5.851172 MULTIPOINT ((-4.041924 51.0...
## 4   9.554778  9.554778  4.439580  4.439580 MULTIPOINT ((-5.928833 54.5...
## 5  13.283601 13.283601 11.002667 11.002667 MULTIPOINT ((-1.875024 52.4...
## 6  10.227034 10.227034 10.909861 10.909861 MULTIPOINT ((-1.829999 52.4...
## 7  11.189479 11.189479 11.002667 11.002667 MULTIPOINT ((-1.830583 52.5...
## 8  12.487220 12.487220 11.002667 11.002667 MULTIPOINT ((-1.830861 52.5...
## 10 11.660835 11.660835  9.966563  9.966563 MULTIPOINT ((-2.584482 51.4...
## 11 15.020310 15.020310 12.311250 12.311250 MULTIPOINT ((-0.175269 51.5...
##    pred_mean   pred_ll   pred_ul
## 1   6.610622  4.816165  8.216905
## 2   6.402592  3.901235  8.230692
## 3   9.135266  7.530268 11.008256
## 4   8.107924  6.581116  9.981859
## 5  11.764932 10.389277 13.155441
## 6  11.499640 10.374513 12.632421
## 7  11.654478 10.481280 12.835725
## 8  11.656008 10.480792 12.839628
## 10 11.081951  9.498505 12.771989
## 11 13.527924 12.040645 14.992659
##
## [[2]]
## Simple feature collection with 92 features and 5 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: -8.05 ymin: 50.05 xmax: 1.55 ymax: 60.85
## geographic CRS: GCS_WGS_84_with_axis_order_normalized_for_visualization
## First 10 features:
##       value   avalue pred_mean  pred_ll  pred_ul                       geometry
## 1  1.391462 1.391462  5.871256 3.168847 8.524835 POLYGON ((-1.45 60.85, -0.8...
## 2        NA       NA        NA       NA       NA POLYGON ((-3.25 59.65, -2.6...
## 3        NA       NA        NA       NA       NA POLYGON ((-3.25 59.05, -2.6...
## 4  1.550536 1.550536  6.095000 3.811810 8.691070 POLYGON ((-6.85 58.45, -6.2...
## 5  1.817049 1.817049  6.155144 4.008645 8.491361 POLYGON ((-5.05 58.45, -4.4...
## 6  1.937049 1.937049  6.183671 4.013364 8.481687 POLYGON ((-4.45 58.45, -3.8...
## 7  1.747014 1.747014  6.254722 4.034984 8.857751 POLYGON ((-7.45 57.85, -6.8...
## 8  2.018250 2.018250  6.315592 4.236477 8.657813 POLYGON ((-5.65 57.85, -5.0...
## 9  2.136083 2.136083  6.334508 4.301313 8.560375 POLYGON ((-5.05 57.85, -4.4...
## 10 2.587760 2.587760  6.455160 4.421099 8.463629 POLYGON ((-3.85 57.85, -3.2...
```

```r
ggplot(data = boundaryregion) + geom_sf() +
  geom_sf(data = respre[[1]], aes(geometry = geometry, color = pred_mean))+
  labs(title = "Average PM 2.5  Level 2016, UK", fill = "PM 2.5")
```




```r
ggplot(data = boundaryregion) + geom_sf() +
  geom_sf(data = respre[[2]], aes(geometry = geometry, fill = pred_mean))+
  labs(title = "Average PM 2.5  Level 2016, UK", fill = "PM 2.5")
```



Average PM 2.5 Level 2016, UK
60°N
58°N
56°N
54°N
52°N
50°N
8°W
6°W
4°W
2°W
0°
2°E
PM 2.5
12
10
8
6