BST 6200 Spatial Statistics and Disease Mapping Homework 3

Miao Cai

4/2/2020

Due 5:15 pm, April 15, 2020

The Ph in soil samples taken in the Great Smokey Mountain National Park are in the file SmokeyMountain.eps. The first few lines of this file are shown below.

Easting	Northing	Ph	Elevation
-79.19	-22.53	7.91	0.244
-61.66	-55.39	7.14	0.375
-69.63	-31.93	6.81	0.567
-54.82	-44.38	6.97	0.512
-62.14	-37.94	7.21	0.408
-55.29	-14.60	6.94	0.512

Use the methods of section 6.8 in the book, including

- Nearest neighbor interpolation (using Voronoi polygons)
- Inverse distance weighting (use $\alpha = 0.5$).
- Kriging (in the semivariogram, give an eyeball estimate of the sill, range, and nugget.)

Explain your reasoning and justify your assertions with the appropriate plots. Submit a PDF file to Blackboard, obtained either by using Word with copy & paste from R, or by creating a PDF file in RMarkdown.

Important note: If you follow the book's development on the fulmar data set, notice that fulmar is the name of the data frame and also one of the variables in the data frame. This can be confusing because you can't tell whether you should tell R what the data frame is or which variable in the data from to refer to. Sometimes fulmar would translate to smokeyMountain (if you call your data frame smokeyMountain) and sometimes fulmar would translate to Ph (if you're referring to the outcome variable in the data frame).

1 Nearest neighbor interpolation (using Voronoi polygons)

```r
pacman::p_load(sp, deldir, tmap, gstat)

# Modified Carson Farmer code
voronoipolygons = function(layer) {
  crds <- layer@coords
  z <- deldir(crds[,1], crds[,2])
  w <- tile.list(z)
  polys <- vector(mode='list', length=length(w))
  for (i in seq(along=polys)) {
    pcrds <- cbind(w[[i]]$x, w[[i]]$y)
    pcrds <- rbind(pcrds, pcrds[1,])
    polys[[i]] <- Polygons(list(Polygon(pcrds)),
                          ID=as.character(i))
  }
  SP <- SpatialPolygons(polys)
  voronoi <- SpatialPolygonsDataFrame(SP,
                data=data.frame(x=crds[,1],
                                y=crds[,2],
                                layer@data,
                                row.names=sapply(slot(SP, 'polygons'),
                                          function(x) slot(x, 'ID'))))
  proj4string(voronoi) <- CRS(proj4string(layer))
  return(voronoi)
}
```

Nearest neighbour interpolation estimate the outcome ph at location (x, y) at the observation point closest to (x, y).

```
sm_sp = SpatialPointsDataFrame(d[,.(x = Easting, y = Northing)],
                               d[,.(x = Easting, y = Northing, Ph)])
proj4string(sm_sp) = CRS("+init=epsg:32631")
sm_voro = voronoipolygons(sm_sp)
```

```
##
##      PLEASE NOTE:  The components "delsgs" and "summary" of the
##  object returned by deldir() are now DATA FRAMES rather than
##  matrices (as they were prior to release 0.0-18).
##  See help("deldir").
##
##      PLEASE NOTE: The process that deldir() uses for determining
##  duplicated points has changed from that used in version
##  0.0-9 of this package (and previously). See help("deldir").
```

```
tm_shape(sm_voro) +
  tm_fill(col = 'Ph', title = 'Ph') +
  tm_borders() +
  tm_dots(size = 0.1)
```

2 Inverse distance weighting (use $\alpha = 0.5$)

The *inverse distance weighting* (IDW) approach estimate the outcome ph at location (x, y) with a weighted mean of nearby observations.

```
sample_sm = spsample(sm_voro, type = 'regular', n = 5000)
idw_est = gstat::idw(Ph ~ 1, locations = sm_sp, newdata = sample_sm, idp = 0.5)
```

```
## [inverse distance weighted interpolation]
```

```
tmap_mode("view")
```

```
## tmap mode set to interactive viewing
```

```
tm_shape(idw_est) +
  tm_dots(col = 'var1.pred')
```

```
## Linking to GEOS 3.6.1, GDAL 2.2.3, PROJ 4.9.3
```

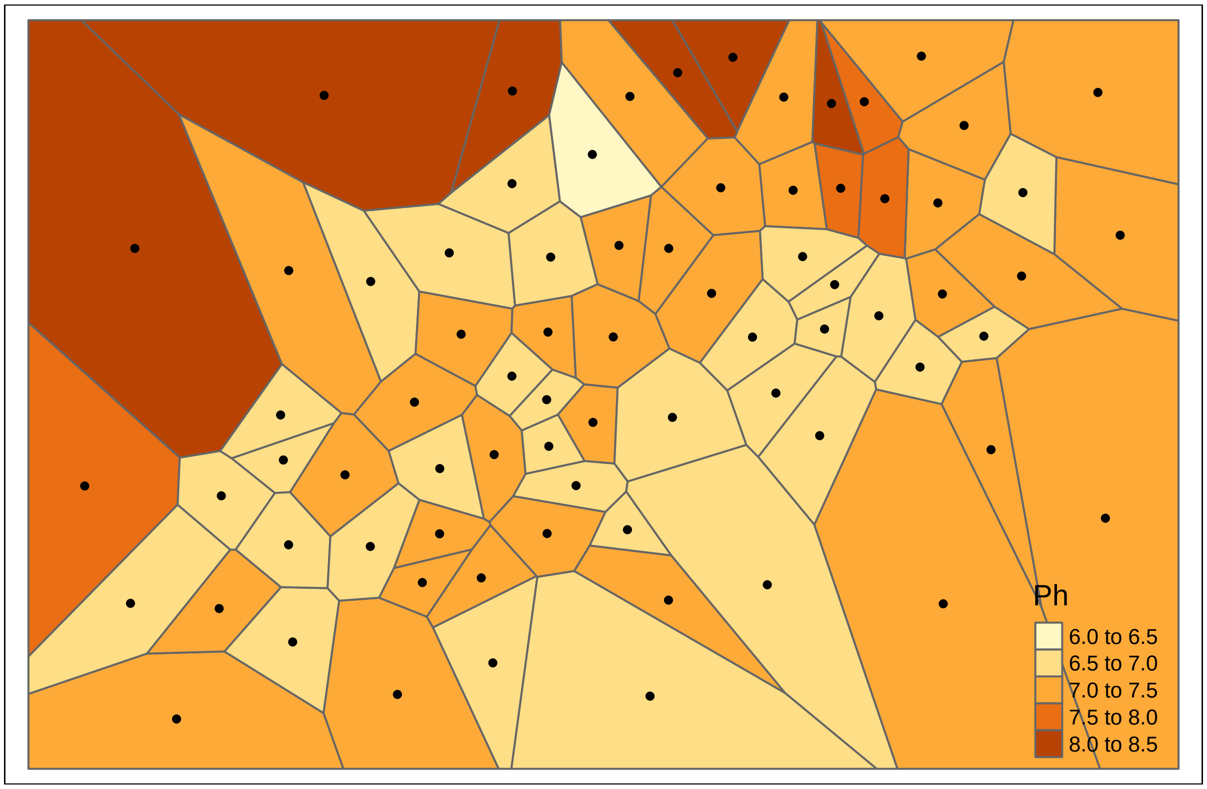
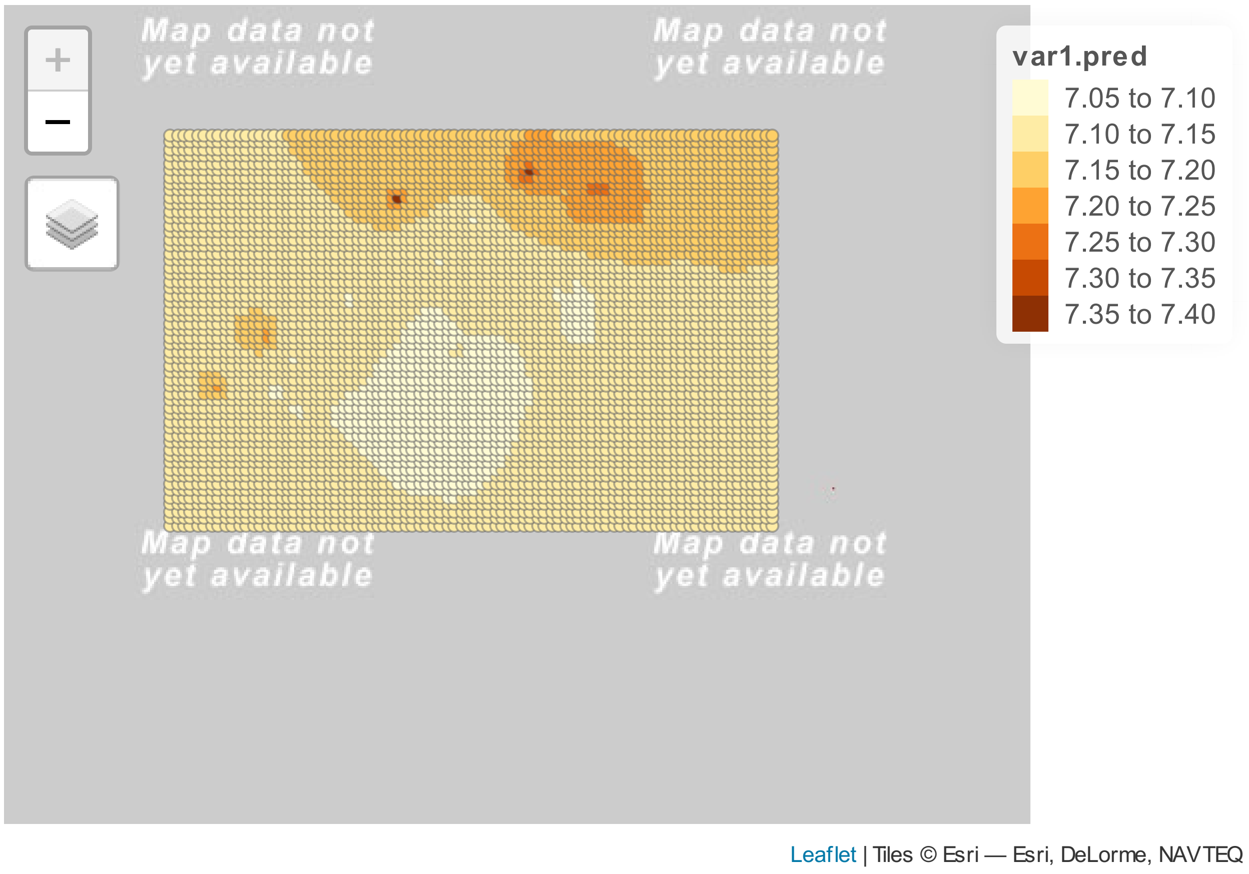


Figure 1: Nearest neighbour interpolation of the smokey mountain data: points and Voronoi diagram



Map data not yet available
Map data not yet available
Map data not yet available
Map data not yet available
var1.pred
7.05 to 7.10
7.10 to 7.15
7.15 to 7.20
7.20 to 7.25
7.25 to 7.30
7.30 to 7.35
7.35 to 7.40
Leaflet | Tiles © Esri — Esri, DeLorme, NAVTEQ

3 Kriging

Kriging defines the outcome z at location x_i using the following function:

$$z(x_i) = f(x_i) + \nu(x_i) + \epsilon_i$$

where $f(x_i)$ is the true function, $\nu(x_i)$ is a random function, and $\epsilon_i \sim$ i.i.d. Normal$(0, \sigma^2)$.

```
sm_evgm = variogram(Ph ~ 1, sm_sp)
sm_fvgm = fit.variogram(sm_evgm, vgm('Mat'))
plot(sm_evgm, model = sm_fvgm)
```

An eyeball estimate of the sill, range, and nugget:

- **sill** (the z value where the semivariogram first becomes flat): 0.19
- **range** (the d value where the semivariogram first becomes flat): 45
- **nugget** (the z-value at a distance of 0): 0.01.

```
sample_sm = spsample(sm_voro, type = 'regular', n = 5000)
krig_est = krige(Ph ~ 1, sm_sp, newdata = sample_sm, model = sm_fvgm)
```

```
## [using ordinary kriging]
```

```
krig_grid = SpatialPixelsDataFrame(krig_est, krig_est@data)
```

```
krig_map_est = tm_shape(krig_grid) +
  tm_raster(col = 'var1.pred', title = 'Ph Density', palette ='Reds') +
```

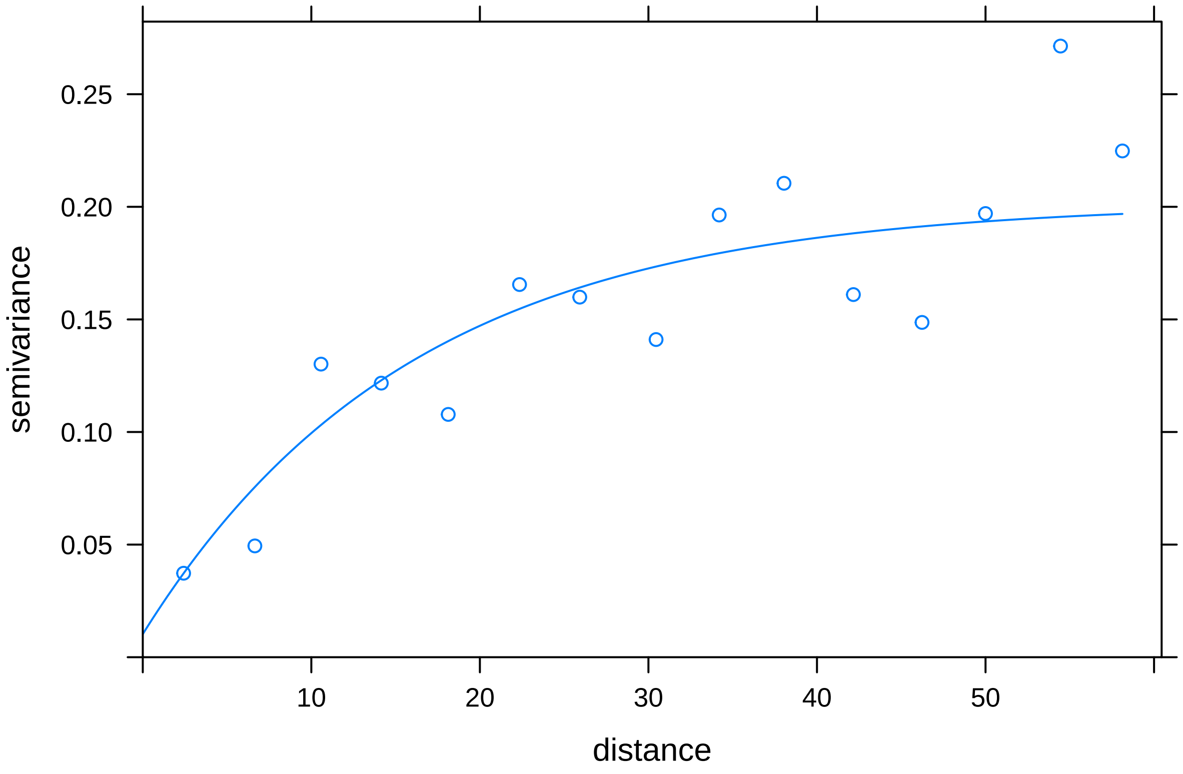


Figure 2: Kriging semivariogram for the Great Smokey Mountain data

```
  tm_layout(legend.bg.color = 'white',legend.frame = TRUE)

krig_map_var = tm_shape(krig_grid) +
  tm_raster(col = 'var1.var', title = 'Estimate Variance', palette='Reds') +
  tm_layout(legend.bg.color = 'white', legend.frame = TRUE)

tmap_arrange(krig_map_est, krig_map_var, ncol = 2)
```

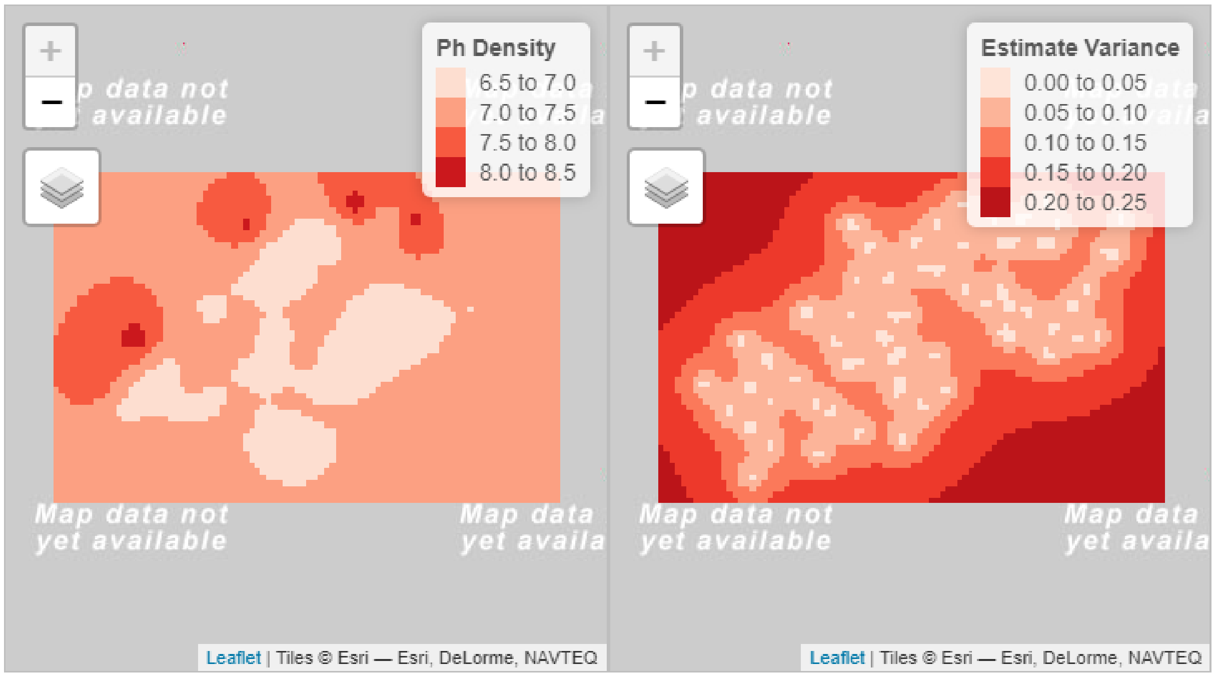


Figure 3: Kriging estimates of Ph density (left) and associated variance (right) for the Great Smokey mountain data