Homework 5

Peter Tran

```
library(GISTools)
library(RColorBrewer)
library(deldir)
library(gstat)
library(sp)
library(tmap)
```

Ok, let's start by pulling in the meuse data. We'll go ahead and create a `SpatialPointsDataFrame` from it, and set the correct projection.

```
data(meuse)
meuse_sp <- SpatialPointsDataFrame(cbind(meuse$x, meuse$y), meuse)
proj4string(meuse_sp) <- CRS("+init=epsg:32631")
```

Let's define the `voronipolygons` function given to us.

```
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)
}
```

Now we'll generate the voronoi polygons for this dataset, and get an idea of the ranges for copper values.

```
meuse_voro <- voronoipolygons(meuse_sp)
```

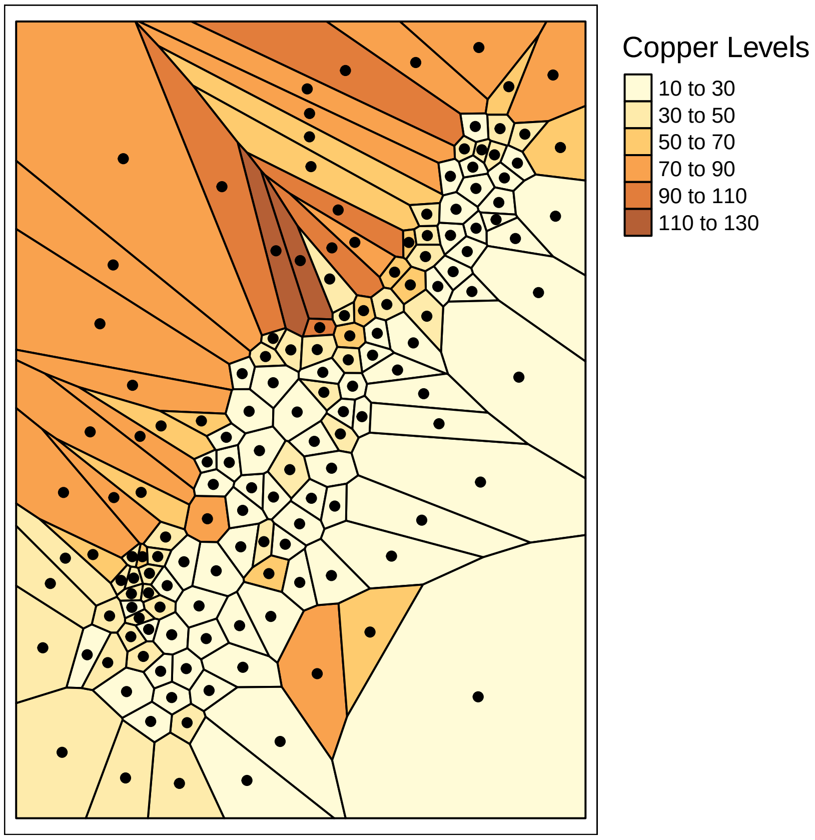
```
## Warning in proj4string(layer): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html
```

```
summary(meuse_voro$copper)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.00   23.00   31.00   40.32   49.50  128.00
```

Now we can actually plot the voronoi polygons.

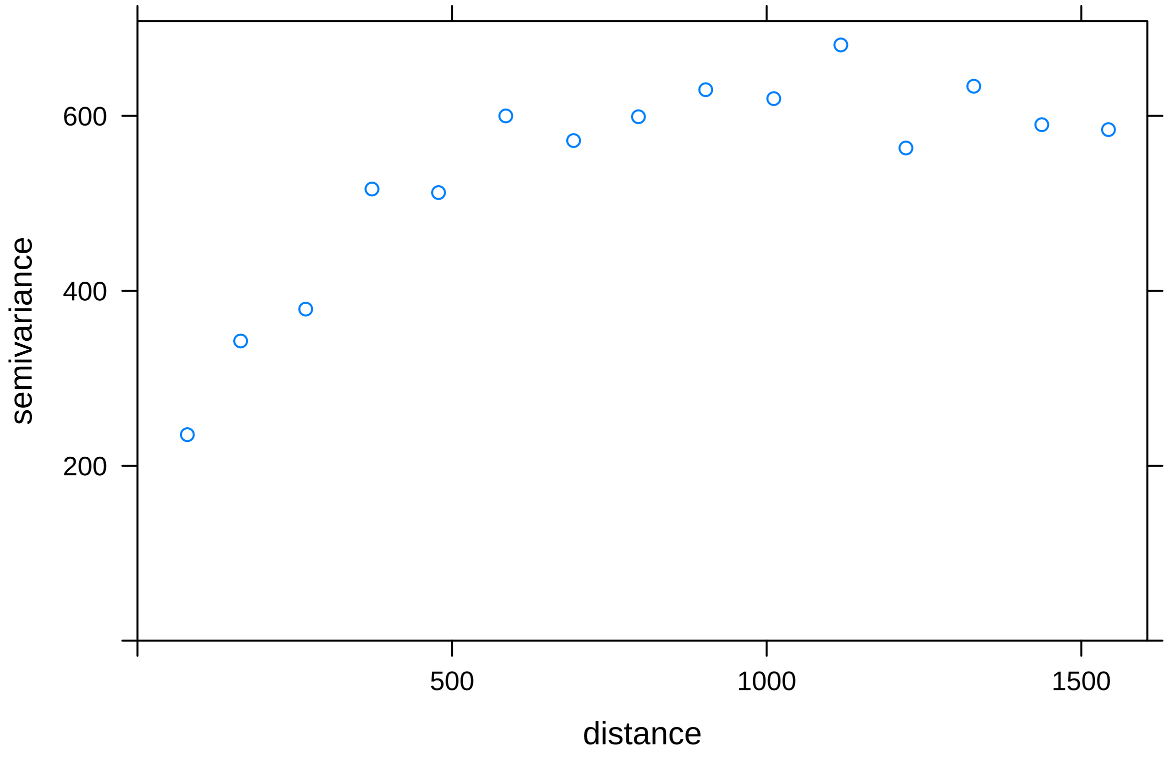
```
meuse_pt <- tm_shape(meuse_sp) +
  tm_dots(size = 0.2)

meuse_vr <- tm_shape(meuse_voro) +
  tm_borders(col = "blue") +
  tm_dots(size = 0.1, col = "red")

shades <- shading(breaks = seq(2, 10, 2), cols = brewer.pal(6, "Reds"))
tm_shape(meuse_voro) +
  tm_fill(col = "copper", style = "fixed",
          breaks = c(10, 30, 50, 70, 90, 110, 130), alpha = 0.8,
          title = "Copper Levels") +
  tm_borders(col = "black") +
  tm_dots(size = 0.1, col = "black") +
  tm_layout(legend.outside = T)
```



Looks like it tells us that we can find copper generally along the northwestern edge of the river in this section, with a moderate deposit in the southeastern region.

Now to try the kriging approach, the first thing we want to do is get the semivariogram.

```r
meuse.vgm <- variogram(meuse$copper ~ 1, meuse_sp)
plot( meuse.vgm )
```



Now we want to fit a smooth function to the variogram, and we need to select a function to do so. Just eyeballing it, it looks like either an exponential or gaussian function would work well here. I'm leaning towards an exponential function, and if I try to draw it in my minds eye it looks like it would have an intercept or "nugget" of 175.

Next we need to determine the weights and make predictions at each point in our grid. We'll go ahead and do that.

```r
meuse.fit.vgm <- fit.variogram(meuse.vgm, vgm("Exp", nugget = 175))

s.grid <- spsample(meuse_sp, type = "regular", n = 4000)
proj4string(s.grid) <- CRS("+init=epsg:32631")
```

```
## Warning in proj4string(obj): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html
```

```r
krig.est <- krige(copper ~ 1, meuse_sp, newdata = s.grid, model = meuse.fit.vgm)
```

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

```r
krig.grid <- SpatialPixelsDataFrame( krig.est , krig.est@data )

summary(krig.grid)
```

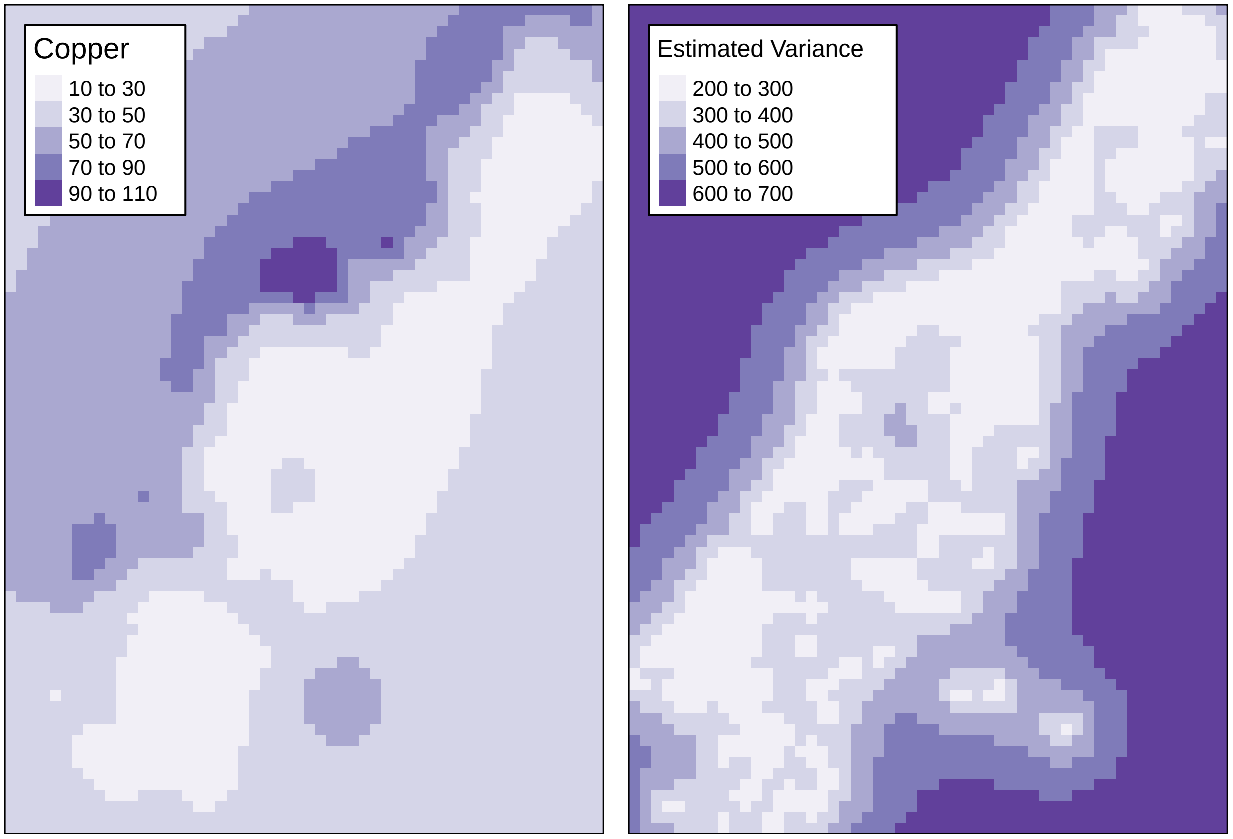
```
## Object of class SpatialPixelsDataFrame
## Coordinates:
##          min      max
## x1 178585.0 181397.9
## x2 329693.9 333600.6
## Is projected: TRUE
## proj4string :
## [+proj=utm +zone=31 +datum=WGS84 +units=m +no_defs]
## Number of points: 4050
## Grid attributes:
##    cellcentre.offset cellsize cells.dim
## x1          178611.1 52.08921        54
## x2          329720.0 52.08921        75
## Data attributes:
##    var1.pred         var1.var
##  Min.   : 16.90   Min.   :195.9
##  1st Qu.: 32.06   1st Qu.:317.4
##  Median : 42.88   Median :504.1
##  Mean   : 45.23   Mean   :481.9
##  3rd Qu.: 55.91   3rd Qu.:644.7
##  Max.   :107.07   Max.   :682.7
```

Ok, we can see our range for the predicted copper values, and also the variance, which is extremely high. This will give us an idea of how to set our breaks.

Now let's plot both the predicted values for the grid as well as the variance for the grid.

```
levs <- c(10, 30, 50, 70, 90, 110)
krig.map.est <- tm_shape( krig.grid ) +
  tm_raster(col = "var1.pred", breaks = levs, title = "Copper", palette = "Purples") +
  tm_layout(legend.bg.color = "white", legend.frame = T)

var.levs <- c(200, 300, 400, 500, 600, 700)
krig.map.var <- tm_shape(krig.grid) +
  tm_raster(col = "var1.var", breaks = var.levs, title = "Estimated Variance",
            palette = "Purples") +
  tm_layout(legend.bg.color = "white", legend.frame = T)

tmap_arrange(krig.map.est, krig.map.var)
```

```
## Warning: Values have found that are less than the lowest break

## Warning: Values have found that are less than the lowest break
```



We see a similar result to the nearest neighbor interpolation, but kriging gives us a little more detail here. We see that most copper is found in the northwestern part of the river, particularly among the more northern parts of this section. We also see that moderate deposit in the southeastern part.

Additionally we can see our variance, which is extremely high all around, but it is lowest in the actual river itself (where most of our measurements are), and lowest the further away from the river you go.