Interpolation via Inverse Path Distance Weighting

Joseph Stachelek

Abstract

The R package **ipdw** provides functions for interpolation of georeferenced point data via Inverse Path Distance Weighting. Useful for coastal marine applications where barriers in the landscape preclude interpolation with Euclidean distances. This method of interpolation requires significant computation and is only practical for relatively small and coarse grids. The **ipdw** implementation may provide additional flexibility and greater speed relative to alternatives.

Keywords: geospatial analysis, connectivity, least-cost path, least-cost distance, R.

1. Introduction

This vignette describes **ipdw**, an R package which provides the functionality to perform interpolation of georeferenced point data using inverse path distance weighting (Suominen *et al.* 2010). Interpolation is accomplished in two steps. First, path distances are calculated from each georeferenced (measurement) point to each prediction point. Path distances, which honor barriers in the landscape, are calculated based on cell-to-cell movement through an underlying `Raster` object (Hijmans 2014) that represents movement cost. These path distances are subsequently used as interpolation weights. The two-step routine follows the order of operations described in Suominen *et al.* (2010) substituting the ESRI path distance algorithm (Mitchell 2012) with the **gdistance** (van Etten 2014) wrapped version of the **igraph** (Csardi and Nepusz 2006) adjacency algorithm.

The **ipdw** package was developed with coastal marine applications in mind where path distances (as the fish swims) rather than Euclidean (as the crow flies) distances more accurately represent spatial connectivity (Little *et al.* 1997). Interpolation of sparse grids in coastal areas otherwise end up bleeding through land areas. The remainder of this vignette provides an example of such a situation using the Kattegat salinity dataset (Diggle and Lophaven 2006) found within the **geoR** package.

2. Data Preparation

To begin, we load the Kattegat dataset. This dataset contains several slots representing the coordinates of point data, the data values, and polylines representing portions of the Kattegat coastline.

```
R> library("ipdw")
R> library("geoR")
R> data(kattegat)
R> katproj<-c("+proj=utm +zone=32 +ellps=GRS80 +units=m +no_defs")
R> pols1<-Polygons(list(Polygon(kattegat$dk[1])),"pol1")
R> pols2<-Polygons(list(Polygon(kattegat$dk[2])),"pol2")
R> pols3<-Polygons(list(Polygon(kattegat$dk[3])),"pol3")
R> pols4<-Polygons(list(Polygon(kattegat$dk[4])),"pol4")
R> pols5<-Polygons(list(Polygon(kattegat$dk[5])),"pol5")
R> pols6<-Polygons(list(Polygon(kattegat$dk[6])),"pol6")
R> pols7<-Polygons(list(Polygon(kattegat$dk[7])),"pol7")
R> pols8<-Polygons(list(Polygon(kattegat$dk[8])),"pol8")
R> pols9<-Polygons(list(Polygon(kattegat$dk[9])),"pol9")
R> pols10<-Polygons(list(Polygon(kattegat$dk[10])),"pol10")
R> pols11<-Polygons(list(Polygon(kattegat$dk[11])),"pol11")
R> pols12<-Polygons(list(Polygon(kattegat$dk[12])),"pol12")
R> pols<-SpatialPolygons(list(pols1,pols2,pols3,pols4,pols5,pols6,
+                         pols7,pols8,pols9,pols10,pols11,pols12),1:12)
```

Each polyline is converted seperately to an object of class `Polygons`. Next, all 12 objects are combined into a `SpatialPolygons` object. We can use this `Polygons` object to create a cost raster defining travel through land areas with a very high cost. As a result, interpolation neighborhoods will be defined based on in-water distances. Cost raster creation is accomplished with the **ipdw** function `costrasterGen`. Open water areas have a per unit travel cost of 1 whereas land areas have a per unit travel cost of 10,000. Note that a projection is defined for the `costrasterGen` function by the `projstr` parameter.

```
R> costras<-costrasterGen(kattegat$coords,pols,katproj)
R> #insert contiguous barrier
R> costras[160:170,1:80]<-10000
```

In order to evaulate the utility of IPDW, we split the dataset into seperate training and validation datasets. The training dataset is created in a spatially balanced manner by building a grid and randomly selecting one measurement point per grid cell. In the following code block, the size of this grid is defined as 2 times the average distance among measurement points. Average distance is computed using the **spatstat** package (Baddeley and Turner 2005). Random selection is accomplished with the **gdata** function `resample` (Warnes *et al.* 2014). Subsetting the full dataset is not required to run `ipdw`. Alternative means of accessing interpolation errors such as leave-one-out cross validation are in development.

```
R> #find average nearest neighbor
R> library(spatstat)
R> W=owin(range(kattegat$coords[,1]),range(kattegat$coords[,2]))
R> kat.pp<-ppp(kattegat$coords[,1],kattegat$coords[,2],window=W)
R> mean.neighdist<-mean(nndist(kat.pp))
R> #grid building
R> gridsize<-mean.neighdist*2
R> grainscale.fac<-gridsize/res(costras)[1]
R> gridras<-aggregate(costras,fact=grainscale.fac)
R> gridpol<-rasterToPolygons(gridras)
R> gridpol$value<-row.names(gridpol)
R> #spatial join
R> kat.df<-data.frame(kattegat)
R> coordinates(kat.df)<-~x.utm+y.utm
R> projection(kat.df)<-katproj
R> fulldataset.over<-over(kat.df,gridpol)
R> fulldataset.over<-cbind(data.frame(fulldataset.over),data.frame(kat.df))
R> #grid selection
R> set.seed(2)
R> gridlev<-unique(fulldataset.over$value)
R> for(i in 1:length(gridlev)){
+   activesub<-subset(fulldataset.over,fulldataset.over$value==gridlev[i])
+   selectnum<-gdata::resample(1:nrow(activesub),1)
+   if(i==1){
+     training<-activesub[selectnum,]
+   }
+   else{
+     training<-rbind(training,activesub[selectnum,])
+   }
+}
```

Next, we save the training and validation datasets as objects of class `SpatialPointsDataFrame`. Note that the projection of the training and validation datasets matches the cost raster we created previously. Calculations within the **ipdw** package require projected datasets. More about R projections can be found from the PROJ.4 documentation at the Open Source Geospatial Foundation (http://trac.osgeo.org/proj/).

```
R> validate<-fulldataset.over[!(row.names(fulldataset.over) %in% row.names(training)),]
R> xy<-cbind(training$x.utm,training$y.utm)
R> training<-SpatialPointsDataFrame(xy,training)
R> xy<-cbind(validate$x.utm,validate$y.utm)
R> validate<-SpatialPointsDataFrame(xy,validate)
R> projection(training)<-katproj
R> projection(validate)<-katproj
```
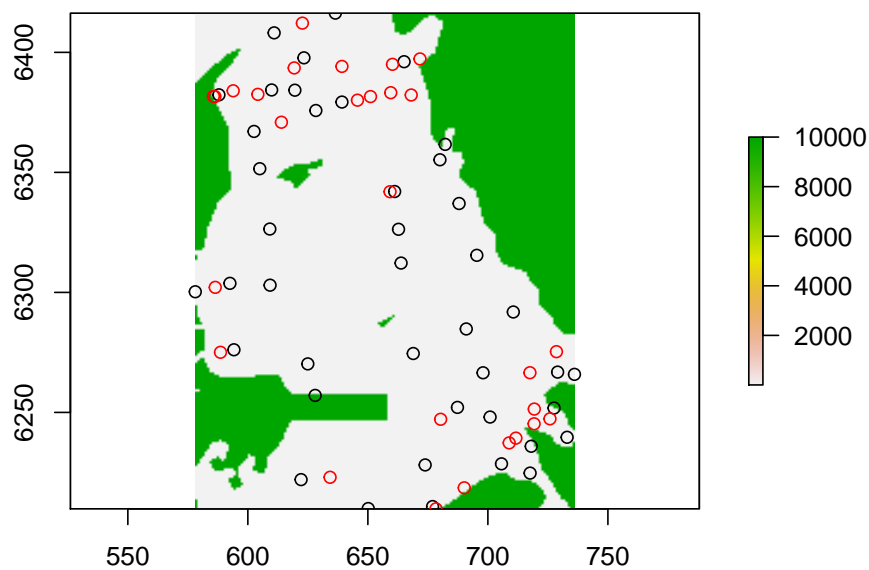

Figure 1: Cost raster representing the high cost of travel through land areas. Training and validation points are shown in black and red respectively.

3. Interpolation

We have assembled an object of class `SpatialPointsDataFrame` to be interpolated and an underlying cost raster of class `Raster`. We can either proceed in a single step using the high-level **ipdw** function `ipdw` or in two steps using calls to the `pathdistGen` and `ipdwInterp` functions. For simplicity, the single step option, `ipdw`, is shown below. The two step option would be useful for the case where we want interpolate multiple parameters of the same `SpatialPointsDataFrame` object using a single `RasterStack` of path distances.

```
R> paramlist<-c("data")
R> final.ipdw<-ipdw(training,costras,range=mean.neighdist*10,paramlist)
```
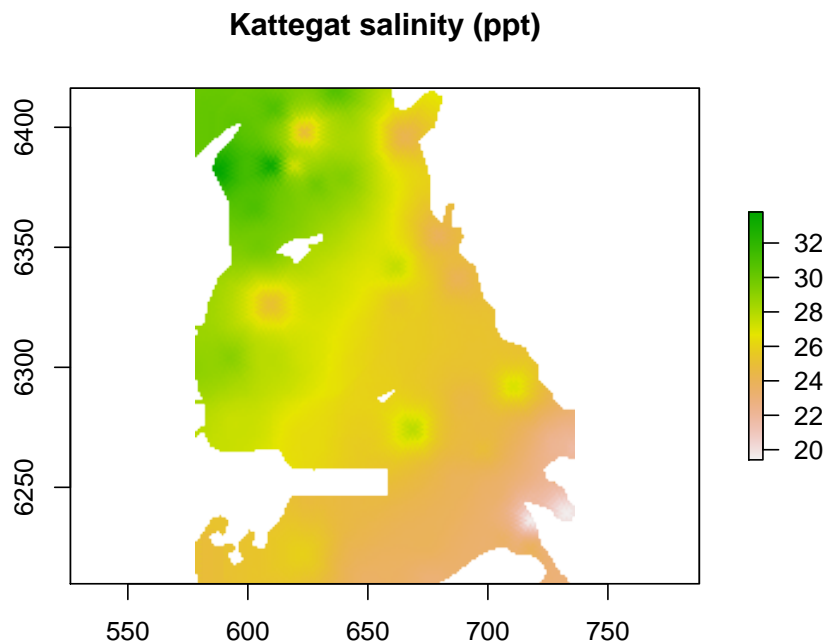
Kattegat salinity (ppt)

Figure 2: Interpolated salinity surface by IPDW.

4. Comparison with Inverse Distance Weighting

The following section generates an interpolated surface via Inverse Distance Weighting (IDW).
First, prediction points are generated. Then the **gstat** (Pebesma 2004) IDW functionality is
called with the same inputs as IPDW in the previous section above.

```
R> idw.grid<-rasterToPoints(costras,fun=function(x){x<10000},spatial=TRUE)
R> gridded(idw.grid)=TRUE
R> kat.idw<-gstat::idw(data~1,training,idw.grid,maxdist=mean.neighdist*10,debug.level=0)
R> final.idw<-raster(kat.idw)
```
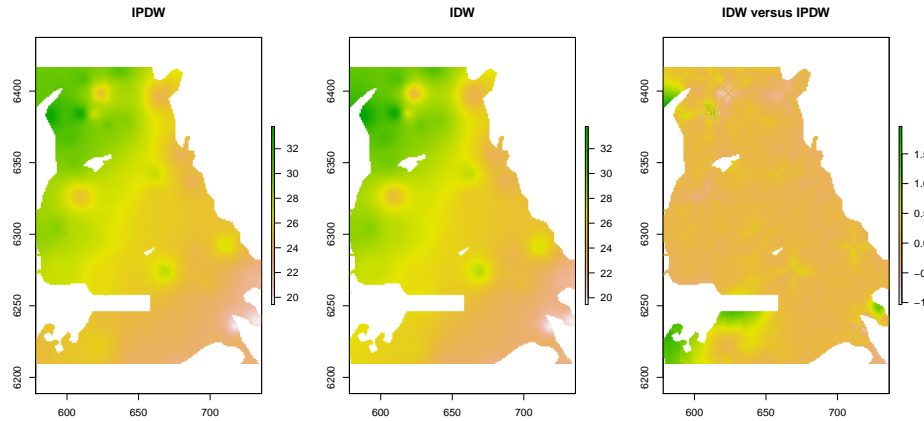
Figure 3: Comparison between IPDW and IDW outputs. Note the overestimation of salinity on the upstream (south) side of the contiguous barrier.

Finally, we can compare our validation dataset with the underlying data in the IPDW and IDW surfaces in order to compare interpolation error. The validation dataset enters into the `errorGen` function both as a `SpatialPointsDataFrame` object and as the underlying values contained in the data slot.

```
R> measured.spdf<-data.frame(validate$data)
R> coordinates(measured.spdf)<-coordinates(validate)
R> valid.ipdw<-errorGen(final.ipdw,measured.spdf,measured.spdf@data)
R> valid.idw<-errorGen(final.idw,measured.spdf,measured.spdf@data)
```
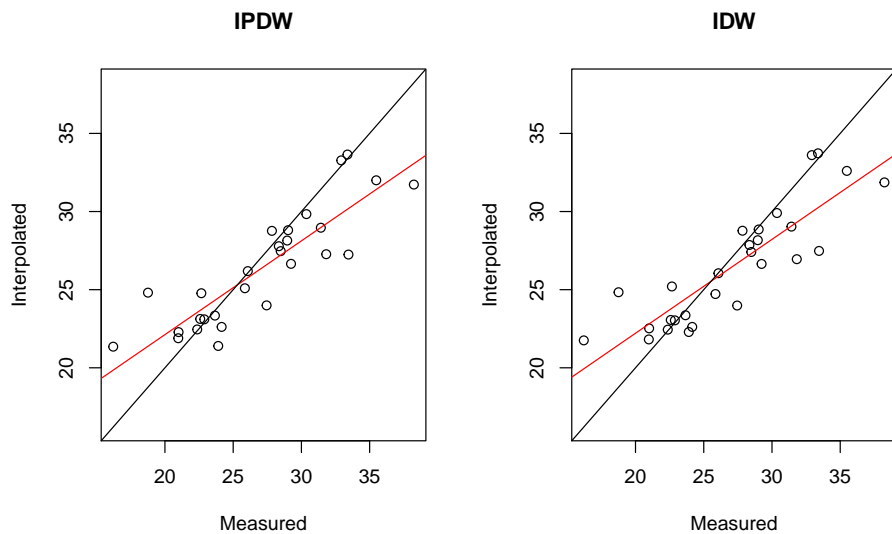

Figure 4: Comparison between IPDW and IDW interpolation error. A one-to-one line and best-fit line are shown in black and red respectively.

Results from IDW and IPDW appear similar because no validation points are present in the area downstream (south) of the contiguous barrier. Up to this point, we have seen a simple implementation of IPDW requiring only a `SpatialPointsDataFrame` and a cost `Raster`.

Test comparisons between the **ipdw** and the ESRI (Mitchell 2012; Suominen *et al.* 2010) implementations of IPDW found **ipdw** to be much faster and more flexible. In particular, the high-level function `ipdw` provided the ability to run IPDW in one step while the lower-level functions `pathdistGen` and `ipdwInterp` could be called multiple times in order to interpolate multiple parameters of a single `SpatialPointsDataFrame`. This is accomplished by saving the output from `pathdistGen`.

References

Baddeley A, Turner R (2005). "spatstat: An R Package for Analyzing Spatial Point Patterns." *Journal of Statistical Software*, **12**(6), 1–42.

Csardi G, Nepusz T (2006). "The **igraph** Software Package for Complex Network Research." *InterJournal, Complex Systems*, **1695**(5).

Diggle P, Lophaven S (2006). "Bayesian Geostatistical Design." *Scandinavian Journal of Statistics*, **33**(1), 53–64.

Hijmans R (2014). *raster:raster: Geographic data analysis and modeling.* R package version 2.2-12, URL http://CRAN.R-project.org/package=raster.

Little L, Edwards D, Porter D (1997). "Kriging in Estuaries: as the Crow Flies or as the Fish Swims?" *Journal of Experimental Marine Biology and Ecology*, **213**.

Mitchell A (2012). *The Esri Guide to GIS Analysis, Volume 3: Modeling Suitability, Movement, and Interaction.* Esri Press.

Pebesma EJ (2004). "Multivariable geostatistics in S: the gstat package." *Computers and Geosciences*, **30**, 683–691.

Suominen T, Tolvanen H, Kalliola R (2010). "Surface Layer Salinity Gradients and Flow Patterns in the Archipelago Coast of SW Finland, Northern Baltic Sea." *Marine environmental research*, **69**(4), 216–226.

van Etten J (2014). *gdistance: distances and routes on geographical grids.* R package version 1.1-5, URL http://CRAN.R-project.org/package=gdistance.

Warnes GR, Bolker B, Gorjanc G, Grothendieck G, Korosec A, Lumley T, MacQueen D, Magnusson A, Rogers J (2014). *gdata: Various R programming tools for data manipulation.* R package version 2.13.3, URL http://CRAN.R-project.org/package=gdata.

Affiliation:

Joseph Stachelek
South Florida Water Management District
West Palm Beach, FL, USA
E-mail: jstachel@sfwmd.gov