Basic Spatial Geometric Calculations

Overview:

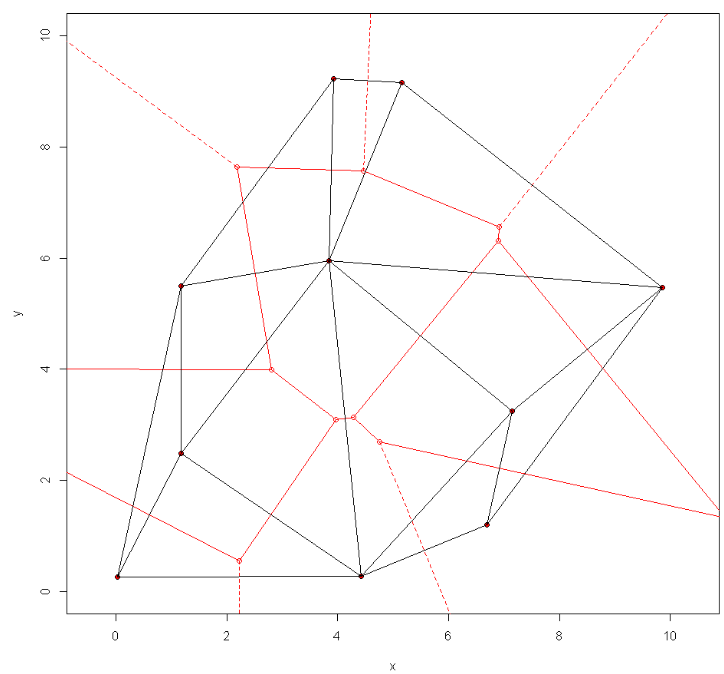
- In spatial analysis we are frequently tasked to represent ***points by areas*** or ***areas by points*** and find a representative point from a set of points.
    - Point to area operations can be achieved with Voronoi polygons
    - Area to point operations lead to several measures of centroids
    - Points to point operations are achieve by weighted means and the weighted Euclidian median
- See also the following topics at online dictionary:
    - Area measure & spherical distance ⇒ http://www.spatialanalysisonline.com/HTML/length_and_area_for_vector_dat.htm
    - Centroids ⇒ http://www.spatialanalysisonline.com/HTML/centroids_and_centers.htm
- Voronoi Polygons, Delaunay Triangulation, and Convex hull
- These methods help to ***transform*** point objects to an associate areal objects.
  However, given a Voronoi polygon the exact location of its underlying generator point cannot be retrieve.
- Use in point pattern analysis: The density of the generator points is ***inversely*** related to the area of the associated Voronoi polygons
- Voronoi polygons are also known as Thiessen tessellations (after the climatologist Thiessen).
- They are mutually exclusive and collectively exhaustive ***areal subdivision*** of the study area base on n ***generator points*** with specific properties:

- Standard Voronoi polygons satisfy the condition that any point within a Voronoi polygon is closer to its generator point than any other generator points.

  $$V(p_i) = \left\{ p \mid \left\| p - p_i \right\| \le \left\| p - p_j \right\| \text{ for } i \ne j \right\}$$

  This defines the ***area of influence*** around the generator points.
- A Voronoi polygon is ***convex*** (i.e., compact). That is, drawing a connecting line from any two points on the edge of the polygon will always remain within the polygon.

- Dual relationship: Voronoi polygons can be derived from Delaunay triangulations and vice versa.
  - In a Delaunay triangulation any ***three closest generator*** points establish a triangulation.
  - The edges of Voronoi polygon cut each edge of the triangulation ***halfway in a right angle***.



- Polygon properties
  - Voronoi polygons around ***exterior*** generator points are theoretically ***unbound***.
    To close exterior polygons, either outside dummy points need to be added at the fringe of the generator points configuration, or an outside circle can be drawn to close the edge polygons.
  - Each ***node*** of a polygon has three ***edges*** (except for degenerated cases)

- Each interior polygon has on average 6 edges (Euler's theorem of planar graphs)
- An edge of a polygon is bisected by its associated triangular segment rectangular.
- Each edge of the polygon is equidistant apart from its two generator points.
- Each node of the polygon is equidistant apart from its three generator points

- Delaunay triangulations can be used to perform deterministic linear interpolation:

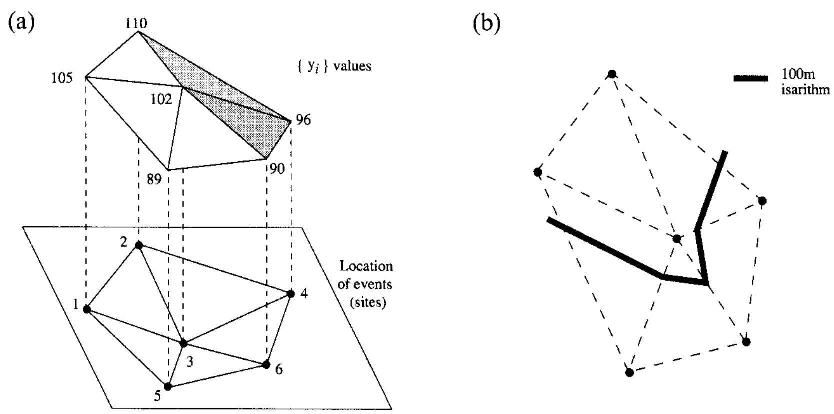
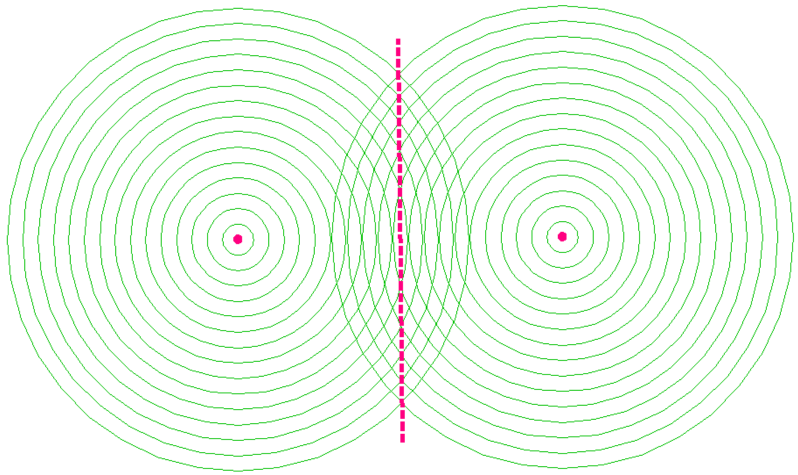


Fig. 5.7 Interpolation using a TIN

- The ***convex hull*** can be built for the outside edges of the triangulation (i.e., by merging all triangles).
- See the see vector-based routines in the R-code `VoronoiPolygons.r`
- A raster-based Voronoi algorithm would check each grid-cell with regards to its proximity to the generator points and assigns the raster cell to the closest generator point.
- Voronoi polygons can also be derived from the perspective of a spatial ***diffusion process***:
  - Waves are extending with equal speed from their equal sized generator points.

- The locations where two wave fronts first touch, determine the polygon boundaries.



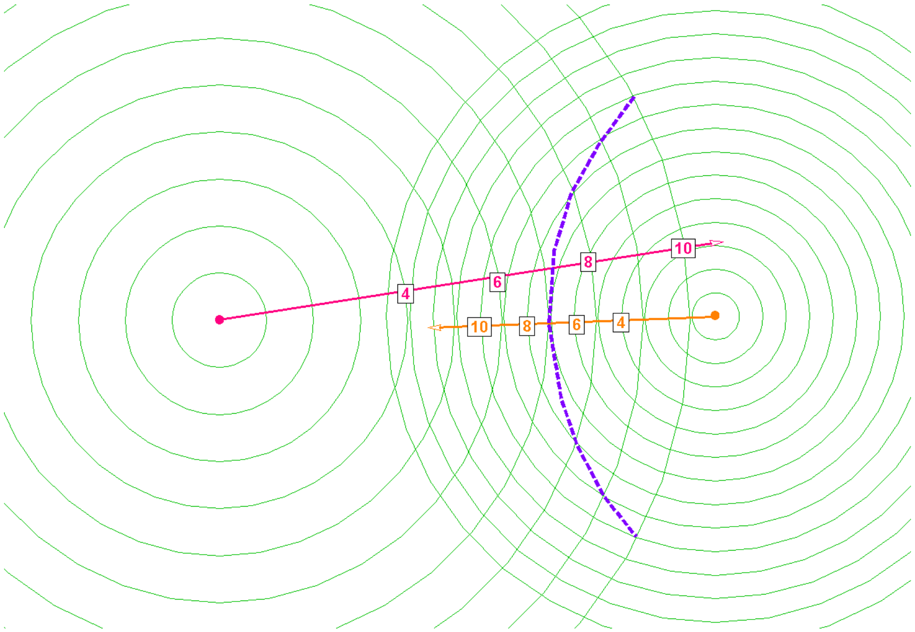
Extensions of the Voronoi Polygon Concept

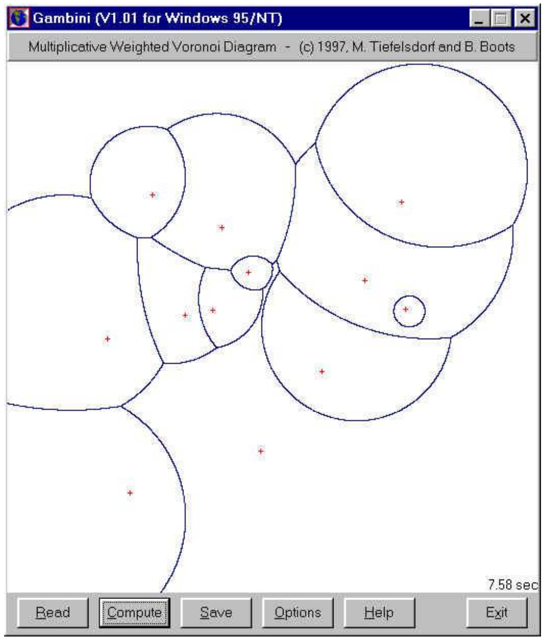
- This diffusion principle can be extended in two ways using ***weighted Voronoi*** polygons:
  - There are no readymade vector-based GIS extensions to calculate weighted Voronoi polygons. Google for “Weighted Voronoi Polygons”.
  - Weighted Voronoi polygons, for instance, have applications in market research assuming a homogeneous underlying geography.

    However, GIS street network algorithms nowadays are becoming more popular. These also can incorporate weights such a speed limits or capacity constraints.

    See http://www.spatialfiltering.com/PreviewGIS/DownLoad/PreviewGISCholeraMaptitude.pdf.

- Waves diffusing with different speeds from the generator points lead to ***multiplicatively weighted*** Voronoi polygons:

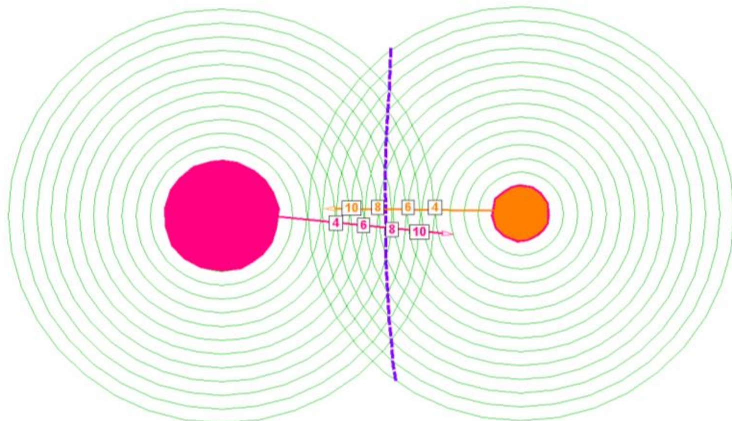




- A multiplicatively weighted Voronoi polygon area around a generator point p_i is defined by

$$V(p_i) = \left\{ p \,|\, \frac{1}{w_i} \cdot \left\| p - p_i \right\| \le \frac{1}{w_j} \cdot \left\| p - p_j \right\| \text{ for } i \ne j \right\}$$

- ○ ***Additively weighted*** Voronoi polygons are conceptionalized as waves diffusing from the generator points at equal speed but with an offset (head start) which is proportional to the weight.



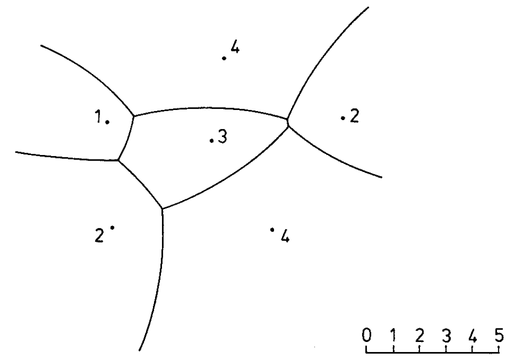


Figure 3.1.5 An additively weighted Voronoi diagram (the numbers indicate weights, w_i).

- ○ An additively weighted Voronoi polygon area around a generator point p_i is defined by

$$V(p_i) = \left\{ p \mid \left\| p - p_i \right\| - w_i \le \left\| p - p_j \right\| - w_j \text{ for } i \ne j \right\}$$

Geographic Central Locations

- ***Polygons can be transformed into representative points***, which may allow calculating distances among polygons.
- Geographic ***arithmetic means*** of a polygon can be derived from point locations within a polygon or from the boundary shape points of a polygon.
- Map projections and the curvature of the earth surface have an influence on the derived locations.

- A standard approach is to calculate the arithmetic means of the easting and northing coordinates of the shape points to get a central location
  - The spatial arithmetic mean is the point $(\bar{x},\bar{y})$ with $\bar{x}=\frac{1}{n}\cdot\sum_{i=1}^{n}x_i$ and $\bar{y}=\frac{1}{n}\cdot\sum_{i=1}^{n}y_i$
  - Recall: The arithmetic mean minimizes the ***squared difference*** between the observations and the central point.

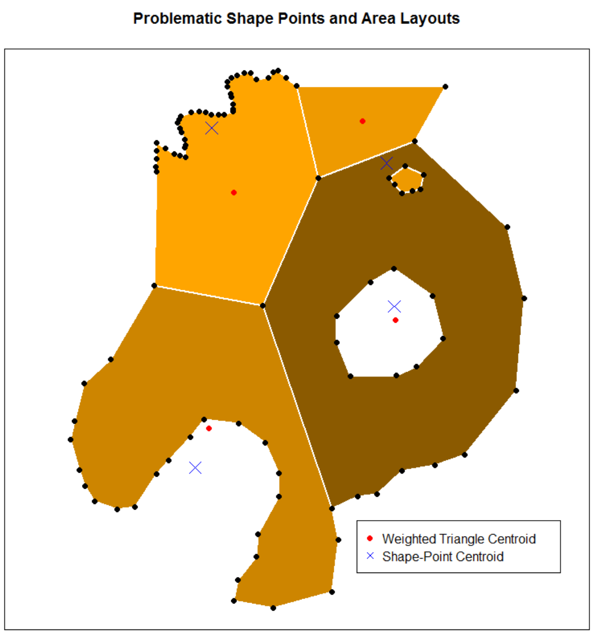
    Thus $(\bar{x},\bar{y})$ ***minimizes the squared distances*** $\sum_{i=1}^{n}d_i^2$ with $d_i=\sqrt{(x_i-\bar{x})^2+(y_i-\bar{y})^2}$.

    The arithmetic mean point is sensitive to outliers (extreme boundary points)
  - However, one must be careful interpreting the representative point of a polygon because:

    [a] Some extreme shape points or a cluster of shape points may have a strong impact on the calculation

    [b] A representative point may not fall into a concave polygon (e.g., a *kidney shaped* polygon, a polygon with a *hole* in it or a polygon that is *fragmented* into several parts).

    Def. ***Concave:*** A line drawn between two arbitrary points within a polygon in parts may be outside the polygon.
  - Example: See code and data of **AREACENTROID.R**



Excurse: Data structure of shape files in R

- The underlying data structure of geographic data is defined in the library **`sp`**.
- Shape-files are stored in an object-oriented framework that inherit properties from subordinate objects:

  [1] an elementary polygon **[Polygon]** are built from lines. These elementary polygons are either holes or areal polygons.

  [2] an aggregated polygon **[Polygons]** is a collection of elementary polygons. Hole polygons are excluded from the aggregated polygon.

  [3] a polygon shape-file **[SpatialPolygons]** is a collection of aggregated polygons.

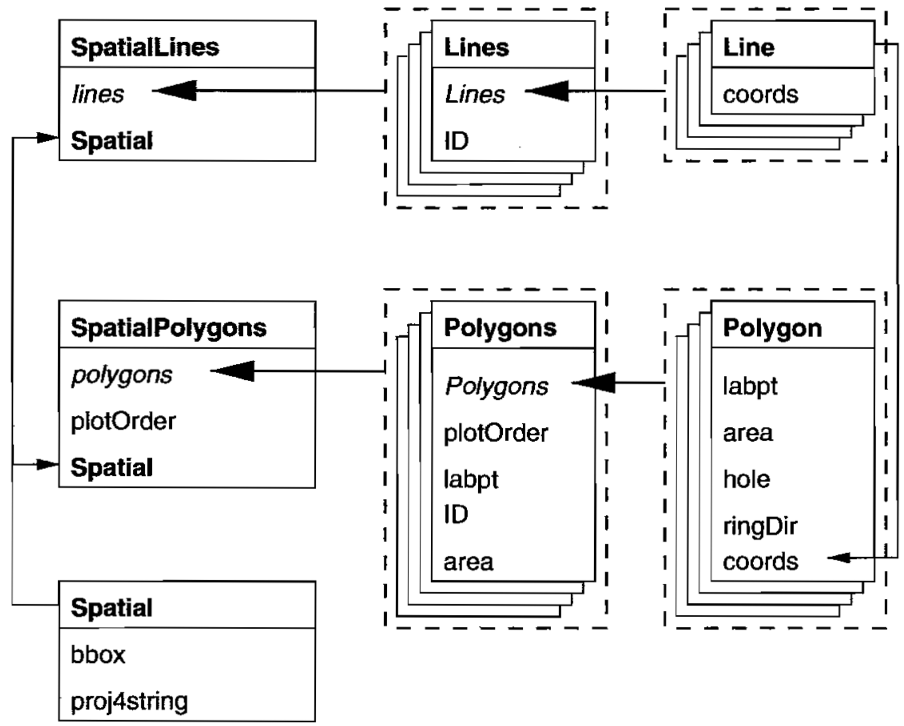


Fig. 2.4. SpatialLines and SpatialPolygons classes and slots; thin arrows show sub-class extensions, thick arrows the inclusion of lists of objects

- Properties of polygon objects are called slots. Slots are similar to fields in list objects, however, instead of addressing a list element with the $-operators, slots are addressed with the @-operator.
- Explore the structure of the **`Shape`** object for shape file **`Region.shp`**.

- Currently, there are activities in the R community to modernize and simplify geography objects. These efforts are led by PEBESMA with "simple feature" file format in the R library **`sf`**. However, most spatial routines in R libraries have not yet been converted to work seamlessly with **`sf`**.
  See https://journal.r-project.org/archive/2018/RJ-2018-009/RJ-2018-009.pdf
- **`sp`** objects can be converted to **`sf`** objects and *vice versa* with
  - ```
    library(sf); library(sp)
    spObj <- as(sfObj, Class="Spatial")
    sfObj <- st_as_sf(spObj, "sf")
    ```

Excurse: Calculation of areas and triangle weighted centroids

- Approximating the area of a polygon by decomposing it into simple geometric shapes and aggregating the areas of these simple shapes:

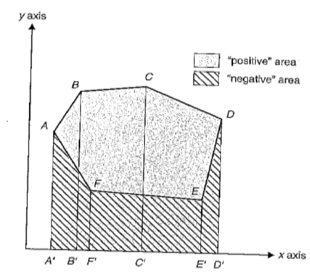


Figure 7.2 Finding the area of a polygon from the coordinates of its vertices.

The area of the simple polygon with the corner points {(A',A) (B',B) (B',0) (A',0)} is calculated by $\frac{1}{2}(B + A) \cdot (B' - A')$.
    ```

- Alternative approach: [a] decompose of a polygon into triangles from a reference point, e.g., $p_0 = (0{,}0)$ and calculate the area of each triangle, [b] add the areas of the triangles together while keeping track of the sign of the sub-areas.

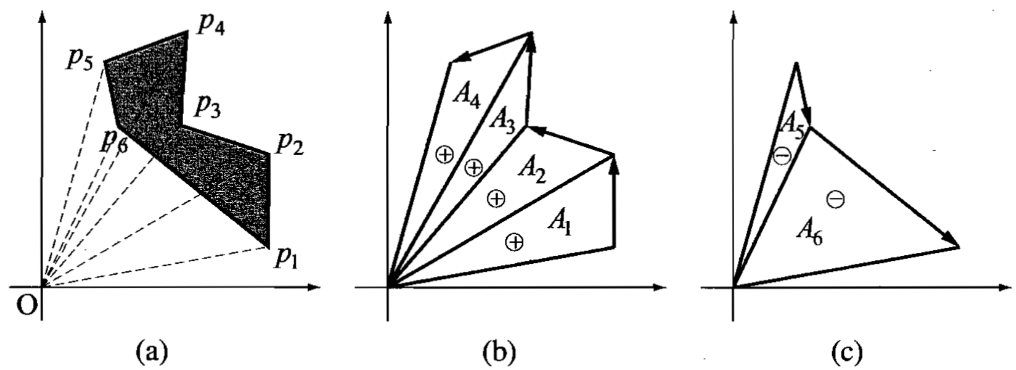


Figure 3.12 Computation of the area of a polygon: (a) a polygon, (b) positively signed areas, (c) negatively signed areas.

- The area of an individual triangle becomes:
- $A_{[p_0,p_1,p_2]} = \frac{y_1+0}{2} \cdot (x_1 - 0) + \frac{y_2+y_1}{2} \cdot (x_2 - x_1) + \frac{0+y_2}{2} \cdot (0 - x_2)$

$$= \frac{1}{2} \cdot (y_1 \cdot x_1 + y_2 \cdot x_2 + y_1 \cdot x_2 - y_2 \cdot x_1 - y_1 \cdot x_1 - y_2 \cdot x_2)$$

$$= \frac{1}{2} \cdot (y_1 \cdot x_2 - y_2 \cdot x_1)$$

- The total area is $A = \sum_{i=1}^{n} A_{[p_0,p_i,p_{i+1}]}$ for a ***closed polygon line***, which is characterized by $p_{n+1} = p_1$.
- The centroid of an individual triangle becomes $\left(\bar{x}_{A_{[p_0,p_1,p_2]}}, \bar{y}_{A_{[p_0,p_1,p_2]}}\right)^T = \left(\frac{x_0+x_1+x_2}{3}, \frac{y_0+y_1+y_2}{3}\right)^T$
- The ***triangle weighted centroid***, which is also the ***center of gravity***, becomes:

$$(\bar{x}_g, \bar{y}_g)^T = \left(\frac{\sum_{i=1}^{n} \left| A_{[p_0, p_i, p_{i+1}]} \right| \cdot \bar{x}_{A_{[p_0, p_i, p_{i+1}]}}}{\sum_{i=1}^{n} \left| A_{[p_0, p_i, p_{i+1}]} \right|}, \frac{\sum_{i=1}^{n} \left| A_{[p_0, p_i, p_{i+1}]} \right| \cdot \bar{y}_{A_{[p_0, p_i, p_{i+1}]}}}{\sum_{i=1}^{n} \left| A_{[p_0, p_i, p_{i+1}]} \right|} \right)^T$$

This is the centroid calculated internally by R and other GIS software systems.

See the script **`GravityCentroidWithArea.R`** using the shape-file **`BowTiePolyShape.shp`**.

Excursion: Medial Axis

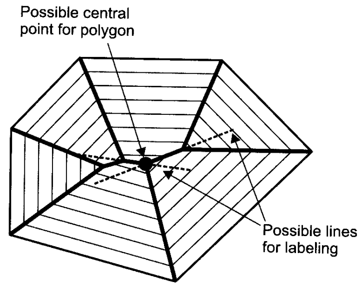
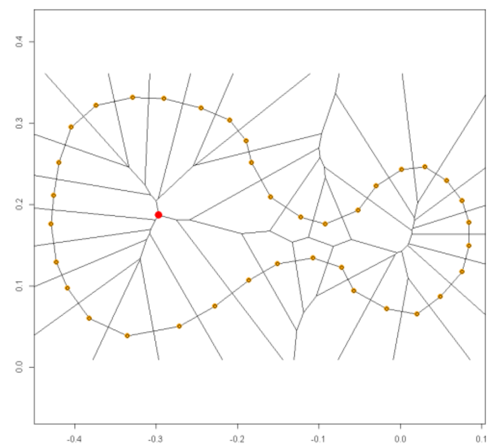


Figure 7.3 Skeleton and resultant center point of a polygon.

- Voronoi polygons can be used to determine the ***skeleton*** or ***medial axis*** of a polygon.

  This reference point has the greatest distance to any boundary point of the polygon.

  The medial axis point will always be located within the interior area of the largest polygon of an area (either convex or concave).
- See **`MedialAxes.R`** using the shape-file **`BowTiePolyShape.shp`**:



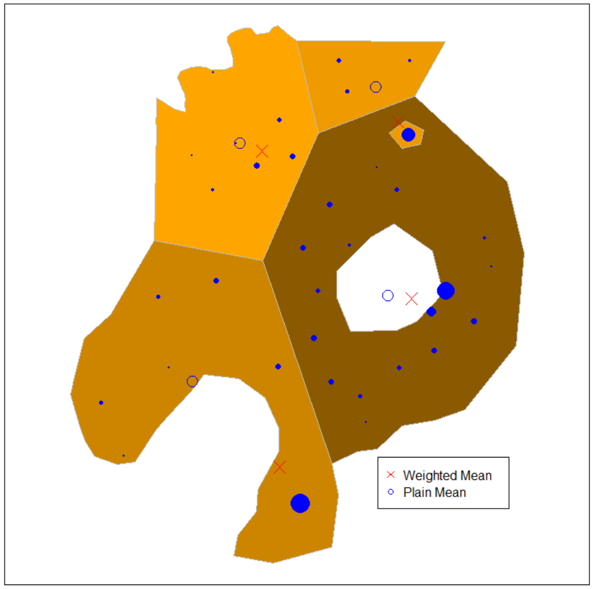
Weighted Arithmetic Mean

- Weighted arithmetic mean centroid takes into account of the ***differential influence of points within a polygon***.
  - It overcomes the homogeneity assumption of constant density of points within a polygon by modeling density variations by within a polygon by weighting its internal points.

    Alternatively, the street network within a polygon can used to model density variations (see, Reibel & Bufalino, 2005. Street-weighted interpolation techniques. *Environment &Planning A*, **37**, 127-139).

- It is defined by $(\bar{x}_w, \bar{y}_w)$ with $\bar{x}_w = \frac{1}{\sum_{i=1}^{n} w_i} \cdot \sum_{i=1}^{n} w_i \cdot x_i$ and $\bar{y}_w = \frac{1}{\sum_{i=1}^{n} w_i} \cdot \sum_{i=1}^{n} w_i \cdot y_i$ and $w_i > 0 \; \forall i$
- If the weights w_i are integer numbers, then weighting scheme can be conceptionalized as replicating a point w_i times in the summation of the standard arithmetic mean.
- For applications in the social sciences, the representative point of a region can be expressed as the population centroid based on the place locations and their population counts.
- See example **WEIGHTEDCENTROID.R**

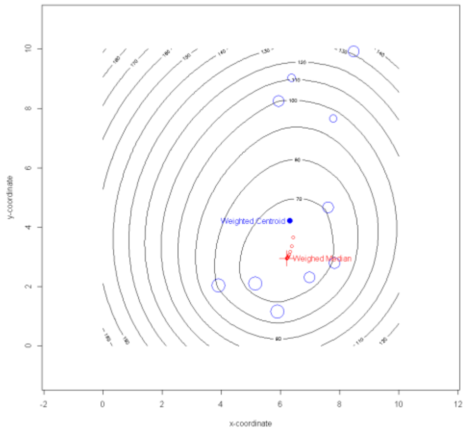
Weighted Regional Means by City Size



Euclidian Median

- The Euclidian median point $(\tilde{x}, \tilde{y})$ overcomes the problem of being sensitive to outliers

- The median center is of relevance in many spatial optimization problems because it minimizes cumulative traveling distances of weighted locations to the median center.
  It ignores however the underlying constraints of a road network.
- Its weighted version $(\tilde{x}_w, \tilde{y}_w)$ minimizes the sum of weighted Euclidian distances to all points, i.e.,
  $\min\limits_{\tilde{x}_w, \tilde{y}_w} \sum_{i=1}^{n} w_i \cdot \sqrt{(x_i - \tilde{x}_w)^2 + (y_i - \tilde{y}_w)^2}$ with $w_i > 0 \ \forall i$
- The unweighted Euclidian median point $(\tilde{x}, \tilde{y})$ is calculated by setting all weights to $w_i = 1 \ \forall i$
- The Euclidian median cannot be calculated analytically but it can be iteratively approximated.
  At each iteration p the tentative median point $(\tilde{x}_p, \tilde{y}_p)$ is updated by weighting observed point locations (x_i, y_i) far from the tentative median point $(\tilde{x}_p, \tilde{y}_p)$ down by using their inverse distance $1/d_{i,p}$ to the tentative median point $(\tilde{x}_p, \tilde{y}_p)$:
  $\tilde{x}_{p+1} = \dfrac{\sum_{i=1}^{n} \frac{w_i \cdot x_i}{d_{i,p}}}{\sum_{i=1}^{n} \frac{w_i}{d_{i,p}}}$ and $\tilde{y}_{p+1} = \dfrac{\sum_{i=1}^{n} \frac{w_i \cdot y_i}{d_{i,p}}}{\sum_{i=1}^{n} \frac{w_i}{d_{i,p}}}$
  See Appendix 3B in Burt and Barber and the code **`EuclidianMedian.r`**

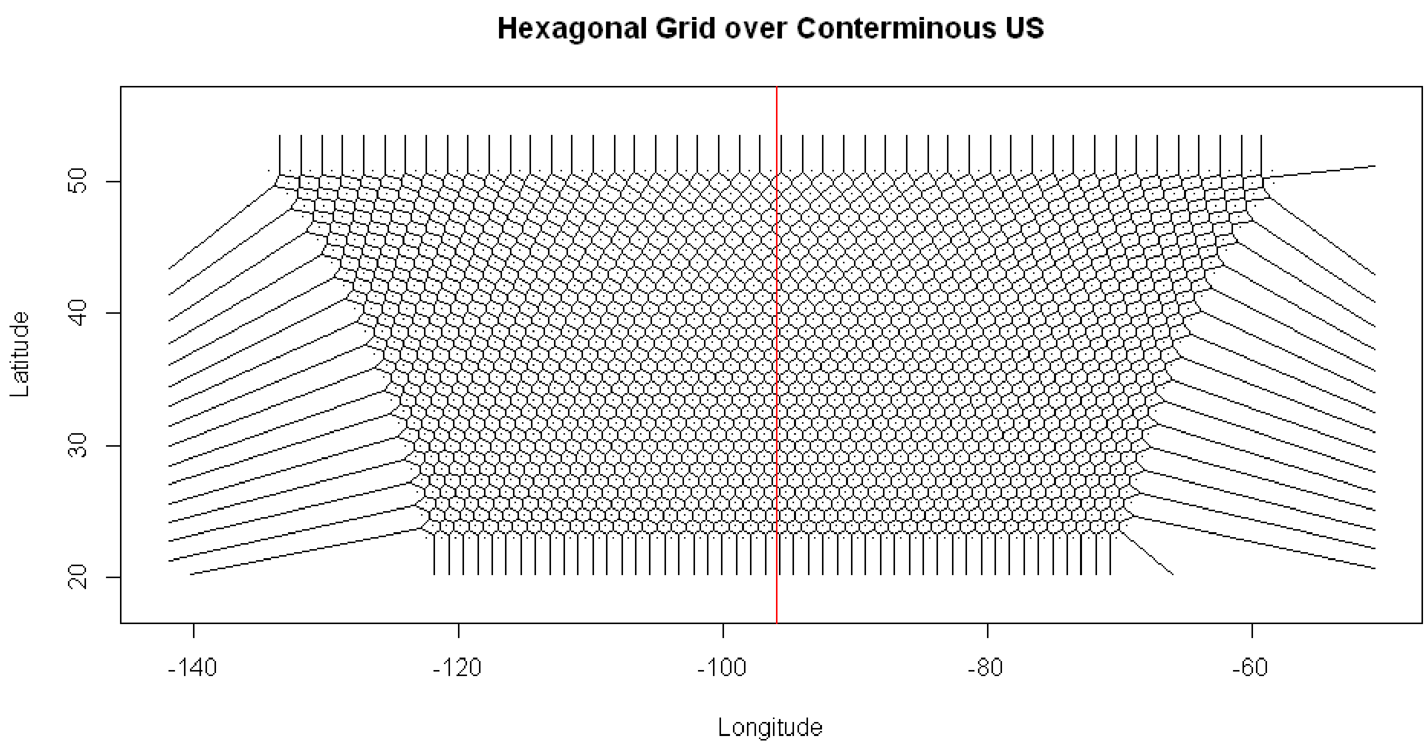



```
Iteration History:
i:  1  x:  6.426  y:  3.649  Cost:  64.02576
i:  2  x:  6.387  y:  3.349  Cost:  63.37127
i:  3  x:  6.338  y:  3.176  Cost:  63.12852
i:  4  x:  6.298  y:  3.078  Cost:  63.04167
i:  5  x:  6.27  y:  3.022  Cost:  63.01109
i:  6  x:  6.25  y:  2.991  Cost:  63.00019
i:  7  x:  6.237  y:  2.973  Cost:  62.99621
i:  8  x:  6.229  y:  2.963  Cost:  62.99472
i:  9  x:  6.223  y:  2.957  Cost:  62.99416
i:  10  x:  6.219  y:  2.954  Cost:  62.99395
i:  11  x:  6.217  y:  2.951  Cost:  62.99386
i:  12  x:  6.216  y:  2.95  Cost:  62.99383
i:  13  x:  6.215  y:  2.949  Cost:  62.99382
i:  14  x:  6.214  y:  2.949  Cost:  62.99381
i:  15  x:  6.214  y:  2.948  Cost:  62.99381
i:  16  x:  6.214  y:  2.948  Cost:  62.99381
i:  17  x:  6.213  y:  2.948  Cost:  62.99381
i:  18  x:  6.213  y:  2.948  Cost:  62.99381

Final Iteration Results:
xMedian =  6.213  yMedian =  2.948 Min. Cost=  62.99381
```

Simple Projected Hexagonal Grid

- See script **`HexaGrid.r`** to a properly projected hexagonal tessellation where irrespectively of the $latitude$ each polygon has an identical area:
  - Define generator points on a triangular grid (equal distance among all points).
  - Center the generator points on a zero meridian $longitude$ of the study area.
  - Re-project the generator points by $1/\cos(latitude)$ to account for shrinking horizontal distances with increasing $latitude$ to maintain equal areas.
  - Shift the generator points back to original meridian $longitude$.
  - Generate Voronoi polygons around the re-projected generator points.
- When this hexagonal tessellation is overlayed over a shapefile in $latitude$ and $longitude$ coordinates each hexagonal cell has the same area.


Hexagonal Grid over Conterminous US
Latitude
20
30
40
50
-140
-120
-100
-80
-60
Longitude