NAVAL
POSTGRADUATE
SCHOOL

OS4118
Statistical and Machine Learning

Spatial Data and Maps

Prof. Sam Buttrey

Fall 2020

The Nation's Premiere Defense Research University

Monterey, California
WWW.NPS.EDU

Spatial Statistics

- My distinction: “spatial statistics” vs. “spatial analysis”
- Spatial **statistics** studies the properties of spatial processes; spatial **analysis** studies real data
  - I claim that real data rarely comes from nicely-defined spatial processes

The Basic Idea

- If there is a spatial component to a process, the values of a process at two points will be more correlated if those two points are close together than if they're far away
- Positive correlation implies attraction, negative correlation implies repulsion

Spatial Processes

- Can be discrete (a “point process”)…
  - E.g. earthquakes, IEDs (“marked PP”)
- …or continuous (a “random field”)
  - E.g. ozone, strength of cell-phone signal
- Without spatial dependence, these can be “completely spatially random” (**CSR**)
  - E.g. two-d homog. Poisson proc. (HPP)
  - If not homogeneous, is it correlation, varying intensity, both, or something else?

Point Process

- We can evaluate whether a process is CSR by measuring *F*- and *G*-functions
  - *G*-function: CDF of distance from one event to its nearest neighbor
  - *F*-function: CDF of distance from arbitrary point to nearest event ("empty space func.")
    - *J*-function: $(1 - G(r))/(1 - F(r))$ is 1 $\forall r$ for HPP
  - Methods for estimating these
  - Compare to values simulated under CSR

Point Process (cont'd)

- Estimate intensity by, e.g. kernel smoothing
- Intensity is a "first-order process"
- "Second order" is correlation
  - Do the points cluster? Is there competition, pushing points away from one another?
  - *K*-function computes number of points within *s* of an arbitrary event
- Old-school approaches like the *K* are being augmented with newer approaches

```
library (sp); library (spatstat)   # Spatial statistics libraries
##
class (japanesepines) # "ppp", i.e. "point pattern in the plane"
#
# A ppp has a matrix-like thing of coordinates and also an
# observation window (of class owin). In general the as() construct
# lets you convert one type to another supported type. E.g.:
#
pppfake    <- as.ppp (spatial.poisson(), owin (c(0 ,20), c(0, 20)))
#
# Example of envelope() command for F-function
#
envelope (pppfake, fun = Fest, nrank=3)
```

Continuous Data

- Instead of incident-type data we have continuous measurements at locations
- **Variogram** measures correlation as a function of distance
  - $\gamma(h) = (1/2)\ \mathbf{E}\{[Z(s) - Z(s+h)]^2\}$…
  - …Under stationarity assumption
  - Under "isotropy," direction doesn't matter; replace *h* with its length

Geographic Data

- Geographic data carries location information
  - Longitude, latitude, maybe altitude and time
  - Alternatives: Military Grid Reference System, Universal Transverse Mercator
- The earth is round-ish, the map is flat
  - Data must be projected onto the plane
  - There are lots of projections, each producing different sorts of distortion
  - Choose one that meets our needs

Maps in R 1: Simple Maps

- The `maps` library makes it straightforward to make simple maps
- Particularly strong on US lower 48 (by county)
- Projections are preserved by default from one call to the next
- Countries appear as sets of named polygons (e.g. France, UK)
- Higher-res data in `library (mapdata)`, `rnaturalearth`, probably others

R Map Examples

- Libraries: `map`, `mapdata`, `mapproj`
- County map of California with Monterey highlighted
- State map of California and Arizona with indications of tunnels
- Examples of projections
  - Path from (0, 90) to (0, 0) to (90, 0) to (90, 60) to (60, 0)
- These are ordinary R graphics, though you will have to project points, lines, text, etc.

Geographic Information Systems

- A GIS is software to hold, display and analyze geographically-referenced data
- GIS is at the intersection of cartography, statistics, and database management
- Our goal: display data in space and time
- Bigger goal: perform analyses of spatial processes
  - Visualization; inference; optimization…

GIS

- Much of the market is held by Esri Corp.'s ArcGIS product, so much of the vocabulary comes from that product
- Many other products exist in different niches (e.g. mapping, waterways, …)
  - Open-source GRASS; also, tools in R (!)
- We have **ArcGIS**, **Google Earth** installed
  - ArcGIS: very powerful, flexible, extensible, but very complicated and hard to learn
  - Google Earth: easy, pretty, links to Google Maps, essentially no analysis

Maps in R 2: Mapping in GIS

- GIS like Google Earth let us combine “layers” of information (photos, roads, geography, waterways, rail, etc.) in a straightforward way
- In GE some of these come automatically
- Google Earth supports only one projection; if our data comes from different sources we may need to re-project some layers, maybe in ArcGIS, to get them to match up
- Main data formats: KML, “Shape files”

Spatial Analysis

- Google Earth is almost entirely for display
- Other GIS perform analyses:
  - "Which polygon contains these points?"
  - "What roads are close to these points?"
  - "Which placement of sub-stations maximizes the number of people within 2,000 m?"
  - "Can this location be seen from that tower?"
- Google Earth is very cool, but it is not a tool for analysis
- Besides ArcGIS, the open-source GRASS is widely used

Format 1: Remember XML?

- XML is a standard mechanism for storing and transporting (but not displaying) data
- It looks like HTML
  - But HTML's focus is on the display
- XML is text-based, so it's not particularly well-suited for floating-point data
- XML isn't quite a language; it's a set of tools for defining a new language

KML

- **KML** ("Keyhole markup language") is the implementation of XML used by Google Earth
  - KMZ: a zipped version that can be read directly
- Google Earth/Maps import and export KML
- KML includes support for
  - Time stamps and animation
  - Altitude and tours
  - Custom data and more

KML Example

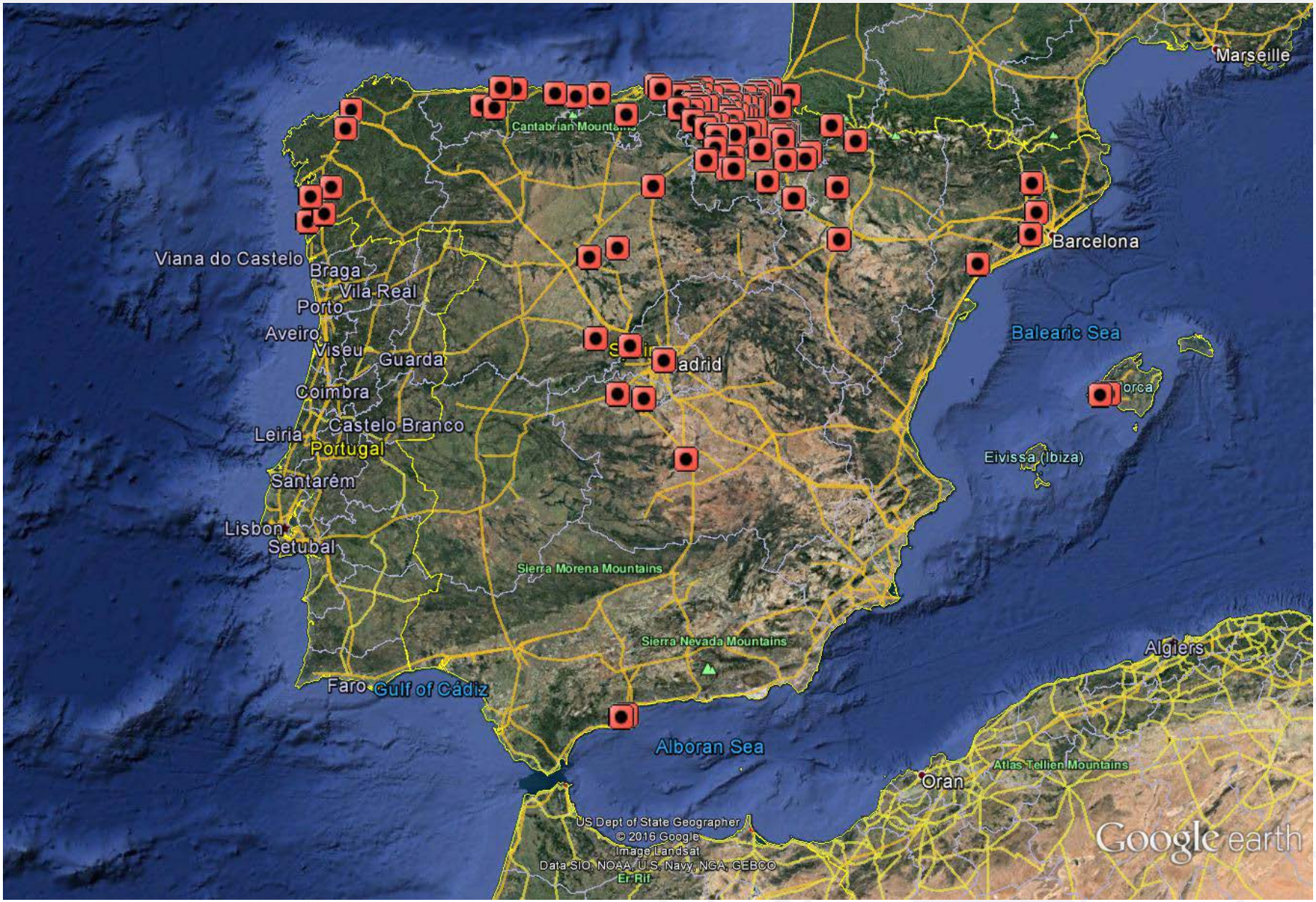
```
<?xml version="1.0“ encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Placemark>
   <TimeStamp>
      <when>2016-09-04T11:00:00-08:00</when>
   </TimeStamp>
<description>Glasgow Hall, NPS</description>
   <Point>
      <coordinates>
        -121.877573,36.59858,0
      </coordinates>
   </Point>
</Placemark>
</kml>
```

Handling KML in R

- `Sp`, `rgdal`, `plotKML`, and some other R packages have some KML facility
- My function will work for now: let's convert the Global Terrorism Database incidents in Spain into KML
- Visualize in Google Earth

NAVAL POSTGRADUATE SCHOOL

Terrorist Incidents in Spain



Google Earth

- Google Earth's servers maintain layers of things like roads and photos, so we get them automatically
- Available in browser or as standalone
- There are servers in the secret labs, too
- Cool examples: Three-d buildings, tours, flight simulator, some underwater mapping…

Alternatives

- Bing Maps / Open Street Map provide nice static maps, **geocoding**
  - Without giving out your credit card number, as Google now requires
  - For “real” products we will need to pay somebody…
  - …By some as-yet undetermined mechanism

Format 2: Shape Files

- This mostly open format started with ArcGIS, now widely used
- A “shapefile” is actually a set of three or more disk files
- Lots available on the web – projection information necessary
- As with Google Earth, a map will be made up of “layers” of different sorts, each with its own display parameters (opacity, point size, line colors, etc.)

Geospatial Data Can Be…

1. Feature Classes
   - Points, lines or polygons with georeferences
2. Attributes

These two are "vector" data

   - Data that describe features: road class, county names, hospital/school/police dept….
3. Image data
   - "Raster data," data in grid/pixel form
   - Can also include non-pictorial continuous data like elevation or other attributes

Shapefiles in R

- Vector shapefile support from:
  - Libraries `sp`, `shapefiles`, `rgdal`
    - "Geospatial data abstraction library"
- Library `raster` for imagery, gridded data
- `rgeos` library for computations like areas, boundaries, centroids, convex hulls…
  - "Geometry engine, open source"
- Three important operations:
  - Reprojection: `spTransform()`
  - Intersection: `over()`
  - Build a buffer: `buffer()`

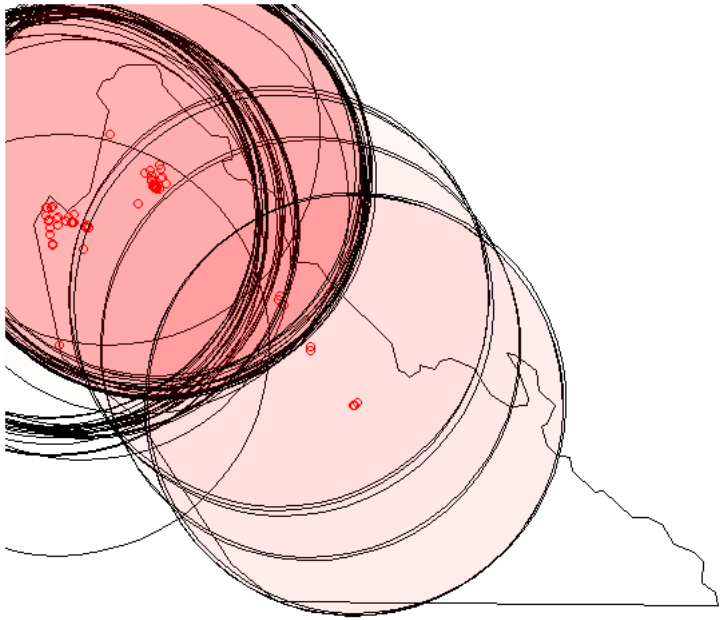
Spatial Analysis Examples

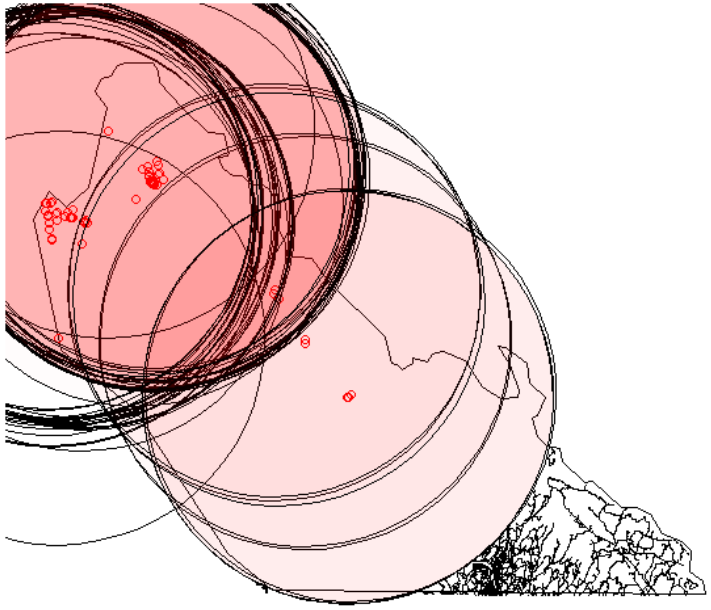
- “Which polygon contains these points?”
- “What roads are close to these points?”
- “Which placement of sub-stations maximizes ‘# of people within 2,000 m’ ?”
- “Can this location be seen from that tower?”
- Ex. 1: “Which roads in Monterey County are > 30 miles from a health care facility?”
- Ex. 2: “What proportion of Malians live within 50 km of a railroad?”

Monterey County Example

- Read (CA health) and (county) shape files
- **Reproject** health to match county
- **Intersect** health with Monterey County
  - Producing health care facilities in county
- **Reproject** health to UTM projection
- **Buffer** facilities to 48,300 m
- Read Monterey roads shapefile
- **Intersect** roads with buffer – find roads that do ***not*** overlap

Pretty Pictures





Mali Example

- Read Rails and Pop Density shape files
- **Reproject** rails to UTM
- **Buffer** rails to 50000 m
- **Intersect** buffer with pop. density map
- Compute ratio of (sum inside buffer) to (sum inside country) – why does this work?