Power of R is Diversity...

... right?

For spatial data (points, lines, polygons, networks... things that have topology) a standard way might be better.

Roger Bivand, author of *Applied Spatial Data Analysis with R*, along with others created sp

Today’s lecture draws heavily from some of his...
The sp package...

- Built using the S4 OO framework
  - Classes
  - “Slots”
  - Methods
  - Inheritance

- Today, how to create, convert, explore and manipulate sp objects

Spatial Data Structures

- Spatial
- CRS (coordinate reference sys.)
Spatial Points

• Most basic of our spatial objects
  – May have coordinates in 2 or 3 dimensions

• Sets of coordinates define a `SpatialPoints` object

• Points in a `SpatialPoints` object may be associated with a row of attributes, a `data.frame` object:
  – `SpatialPointsDataFrame`

Exploring Spatial Points

• Use the Meuse river bank data set of soil samples and measurements of heavy metal pollution provided with `sp`

• Make a `SpatialPoints` object:

```r
> library(sp)
> data(meuse)
> coords <- SpatialPoints(meuse[, c("x", "y")])
> summary(coords)
Object of class SpatialPoints
Coordinates:
   min  max
x 178605 181390
y 329714 333611
Is projected: NA
proj4string : [NA]
Number of points: 155
```
Create `SpatialPointsDataFrame`
We can add the original data frame to make a `SpatialPointsDataFrame` object. Most methods for standard `data.frame` “just work” (e.g. `[]`, `subset()`, etc.)

```r
> meuse1 <- SpatialPointsDataFrame(coords, meuse)
> names(meuse1)
[1] "x" "y" "cadmium" "copper" "lead" "zinc"
[7] "elev" "dist" "con" "ffreq" "soil" "lime"
[13] "landuse" "dist.m"
> summary(meuse1$zinc)
     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.  
   113.0  198.0  326.0   469.7  674.5  1839.0  
> stem(meuse1$zinc, scale = 1/2)
The decimal point is 2 digit(s) to the right of the |
0 | 122233334444455566667778888899999999
2 | 000000011111122222334445556666788880000223344557888
4 | 0001222333677801455555567899
6 | 014446788900124555678889
8 | 01331133
10 | 2356044699
12 | 8
14 | 5357
16 | 7
18 | 4  
```

**Spatial Data Structures**
- `SpatialPoint`
- `SpatialPointsDataFrame`
- `CRS` (coordinate reference sys.)
Spatial points classes and their “slots”

Lines and Polygons

- A Line object is a “spaghetti” collection of coordinates
- A Polygon object is a Line object where the first and last coordinates are identical
  - SpatialLines object is just a list of Line objects
  - SpatialPolygons a list of Polygon objects
- Similarly to Points, we can attach a data.frame and have SpatialLinesDataFrame and SpatialPolygonsDataFrame objects
Polygons Example

- The Meuse river data also include the coordinates of the edge of the river, linked at the edges to form a polygon.

- To make these coordinates into a SpatialPolygons object:
  ```
  > data(meuse.riv)
  > str(meuse.riv)
  num [1:176, 1:2] 182004 182137 182252 182314 182332 ...
  > river_polygon <- Polygons(list(Polygon(meuse.riv)), ID = "meuse")
  > rivers <- SpatialPolygons(list(river_polygon))
  > summary(rivers)
  Object of class SpatialPolygons
  Coordinates:
  min max
  r1 178304.0 182331.5
  r2 325698.5 337684.8
  Is projected: NA
  proj4string : [NA]
  ```

Spatial Lines

- Let’s plot some lines:
  ```
  library(maptools)
  data(volcano)
  image(volcano)
  contour(volcano, add=TRUE)
  ```
Spatial Lines

- Let’s create a `SpatialLinesDataFrame` object:

```r
> library(maptools)
> library(maptools)
> volcano_sl <- ContourLines2SLDF(contourLines(volcano))
> row.names(slot(volcano_sl, "data"))
[1] "C_1" "C_2" "C_3" "C_4" "C_5" "C_6" "C_7" "C_8" "C_9"
[10] "C_10"
> sapply(slot(volcano_sl, "lines"), function(x) slot(x, + "ID"))
[1] "C_1" "C_2" "C_3" "C_4" "C_5" "C_6" "C_7" "C_8" "C_9"
[10] "C_10"
> sapply(slot(volcano_sl, "lines"), function(x) length(slot(x, + "Lines")))
[1] 3 4 1 1 1 2 2 3 2 1
> volcano_sl$level
[1] 100 110 120 130 140 150 160 170 180 190
Levels: 100 110 120 130 140 150 160 170 180 190
```

So the Objects So Far...

```
SpatialLines
    \[lines\]
    \[Spatial\]

Lines
    \[Lines\]
    \[ID\]

Line
    \[coords\]

SpatialPolygons
    \[polygons\]
    \[plotOrder\]
    \[Spatial\]

Polygons
    \[Polygons\]
    \[plotOrder\]
    \[ID\]
    \[area\]

Polygon
    \[labpt\]
    \[hole\]
    \[ringDir\]
    \[coords\]
```

Bivand, 2007
Spatial Grids and Pixels

• Two representations of data on rectangular grids of coordinates
  – SpatialPixels and SpatialGrid
  – Always are oriented N-S, E-W!

• SpatialPixels are like SpatialPoints, but coordinates have to be regularly spaced;
  – Coordinates are stored, but attributes only where there are data

• SpatialGridDataFrame objects do not store coordinates
  – But they do store NA values where attributes are missing

Spatial Pixels

Let’s make a SpatialPixelsDataFrame object for the Meuse bank data; regular points at a 40 m spacing.

```r
> data(meuse.grid)
> coords <- SpatialPixels(SpatialPoints(meuse.grid[, c("x", + "y")]))
> meusegl <- SpatialPixelsDataFrame(coords, meuse.grid)
> names(meusegl)
[1] "x" "y" "part.a" "part.b" "dist" "soil" "ffreq"
> slot(meusegl, "grid")
x y
cellcentre.offset 178460 329620
cellsizes 40 40

cells.dim 78 104
> object.size(meusegl)
[1] 339036
> dim(slot(meusegl, "data"))
[1] 3103 7
```
Spatial Grid Object

We can convert `SpatialPixelsDataFrame` objects directly by coercion. But in most cases we will create a `GridTopology` object first or assign it to a `SpatialPoints/SpatialPixels` object:

```r
> meuseg2 <- meuseg1
> fullgrid(meuseg2) <- TRUE
> slot(meuseg2, "grid")
```

```
x y
cellcentre.offset 178460 329620
cells.size 40 40
cells.dim 78 104
> class(slot(meuseg2, "grid"))
[1] "GridTopology"
attr("package")
[1] "sp"
> object.size(meuseg2)
[1] 425684
> dim(slot(meuseg2, "data"))
[1] 8112 7
```

Spatial Grid and Pixels Classes/Slots

Bivand, 2007
Spatial classes provided by \textit{sp}

This table summarizes all of the classes provided by \textit{sp}, and shows their inheritance/hierarchy structure. They build up to become the objects of most practical use, the \texttt{Spatial*DataFrame} family objects:

<table>
<thead>
<tr>
<th>data type</th>
<th>class</th>
<th>attributes</th>
<th>extends</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td>SpatialPoints</td>
<td>none</td>
<td>Spatial</td>
</tr>
<tr>
<td>points</td>
<td>SpatialPointsDataFrame</td>
<td>data.frame</td>
<td>SpatialPoints</td>
</tr>
<tr>
<td>pixels</td>
<td>SpatialPixels</td>
<td>none</td>
<td>SpatialPoints</td>
</tr>
<tr>
<td>pixels</td>
<td>SpatialPixelsDataFrame</td>
<td>data.frame</td>
<td>SpatialPoints</td>
</tr>
<tr>
<td>full grid</td>
<td>SpatialGrid</td>
<td>none</td>
<td>SpatialPixels</td>
</tr>
<tr>
<td>full grid</td>
<td>SpatialGridDataFrame</td>
<td>data.frame</td>
<td>SpatialGrid</td>
</tr>
<tr>
<td>line</td>
<td>Line</td>
<td>none</td>
<td>Line list</td>
</tr>
<tr>
<td>lines</td>
<td>Lines</td>
<td>none</td>
<td>Line list</td>
</tr>
<tr>
<td>lines</td>
<td>SpatialLines</td>
<td>none</td>
<td>Spatial, Lines list</td>
</tr>
<tr>
<td>lines</td>
<td>SpatialLinesDataFrame</td>
<td>data.frame</td>
<td>SpatialLines</td>
</tr>
<tr>
<td>polygon</td>
<td>Polygon</td>
<td>none</td>
<td>Line</td>
</tr>
<tr>
<td>polygons</td>
<td>Polygons</td>
<td>none</td>
<td>Polygon list</td>
</tr>
<tr>
<td>polygons</td>
<td>SpatialPolygons</td>
<td>none</td>
<td>Spatial, Polygons list</td>
</tr>
<tr>
<td>polygons</td>
<td>SpatialPolygonsDataFrame</td>
<td>data.frame</td>
<td>SpatialPolygons</td>
</tr>
</tbody>
</table>

Bivand, 2007

Some methods provided by \textit{sp}

This table summarizes the methods used to create and manipulation \textit{sp} objects:

<table>
<thead>
<tr>
<th>method</th>
<th>what it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>select spatial items (points, lines, polygons, or rows/cols from a grid) and/or attributes variables</td>
</tr>
<tr>
<td>$&lt;-</td>
<td>retrieve, set or add attribute table columns</td>
</tr>
<tr>
<td>$spsample</td>
<td>sample points from a set of polygons, on a set of lines or from a gridded area</td>
</tr>
<tr>
<td>\texttt{bbox}</td>
<td>get the bounding box</td>
</tr>
<tr>
<td>\texttt{proj4string}</td>
<td>get or set the projection (coordinate reference system)</td>
</tr>
<tr>
<td>\texttt{coordinates}</td>
<td>set or retrieve coordinates</td>
</tr>
<tr>
<td>\texttt{coerce}</td>
<td>convert from one class to another</td>
</tr>
<tr>
<td>\texttt{overlay}</td>
<td>combine two different spatial objects</td>
</tr>
</tbody>
</table>

Bivand, 2007
What now?

- `sp` also provides methods for visualizing spatial data:
  
  [http://r-spatial.sourceforge.net/gallery/](http://r-spatial.sourceforge.net/gallery/)

- `maptools` package also provides further visualization and data options

- `rgdal (e.g. readOGR() and writeOGR()) can use common file formats`
Code to Plot “Bubbles” on top of Polygon Objects

```r
# Plot the spatial subset:
bubble(crossed_pond_chars, "model_rsq",
scales=list(draw=FALSE),
col=c("chartreuse","chartreuse"),
key.entries=seq(from=-0.0, to=1.0, by=0.25),
main = "Pond R-Squared From Cross-Validated Model",
sp.layout =
    list(pond_view, north_arrow, scale_bar, text1),
as.table = TRUE,
xlim=c(463300, 465200),
ylim=c(3005900, 3008000),
pretty=TRUE,
maxsize=3,
alpha=0.50)
```