LANCASTER UNIVERSITY

Geospatial Data in R *And Beyond!*

Barry Rowlingson
b.rowlingson@lancaster.ac.uk

School of Health and Medicine,
Lancaster University

CHICAS
combining health information, computation and statistics

LANCASTER UNIVERSITY

Historical perspective

CHICAS
combining health information, computation and statistics

Before R

- The S Language (Chambers, Becker, Wilks)
  - Born 1976
  - Released 1984
  - Reborn 1988 (The New S Language)
- S-Plus
  - Commercialised version, 1988

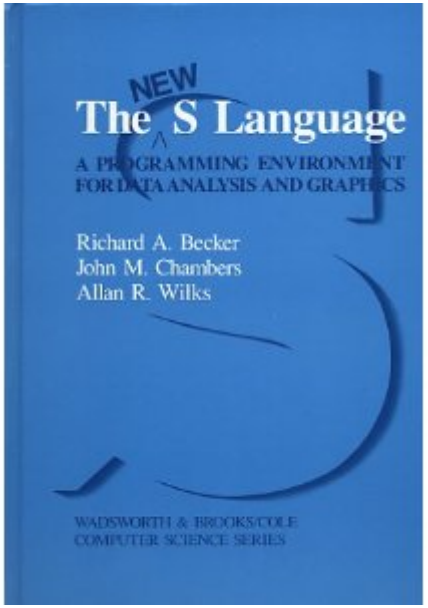
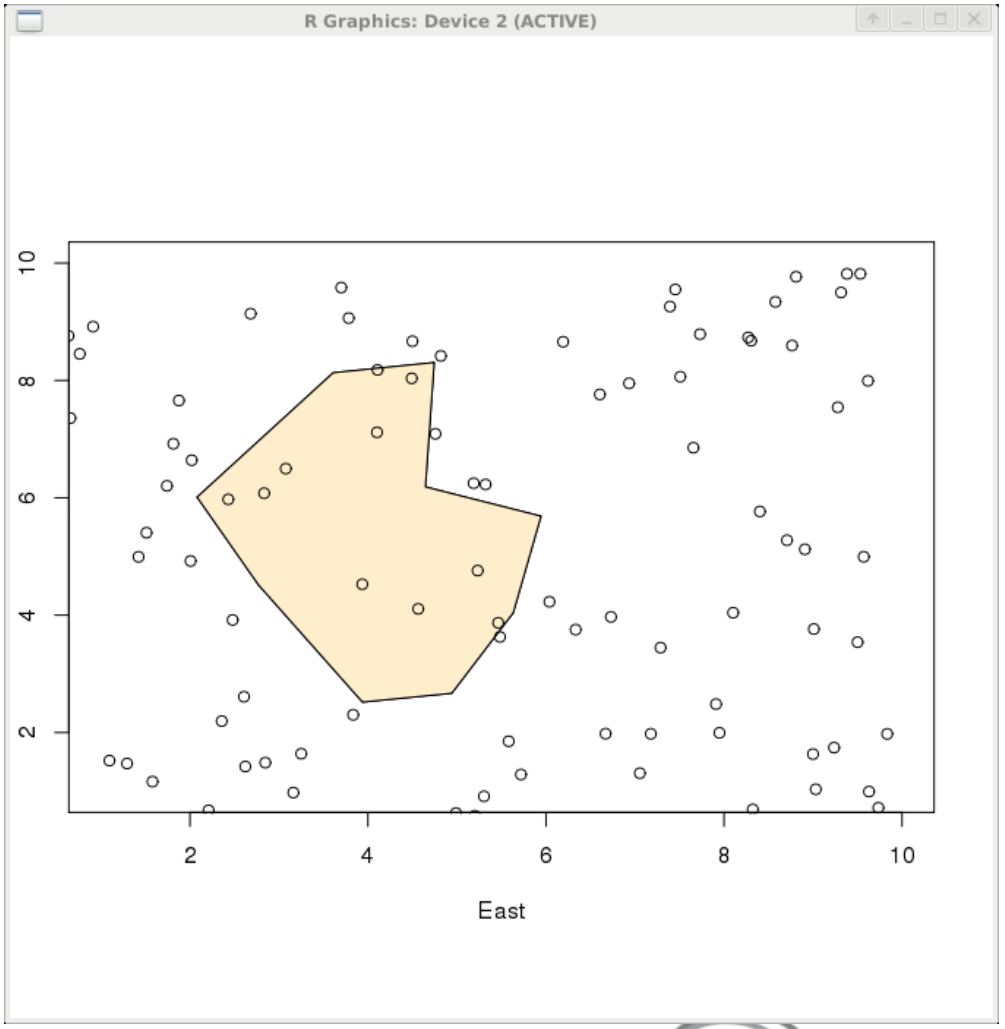


Image from amazon.com

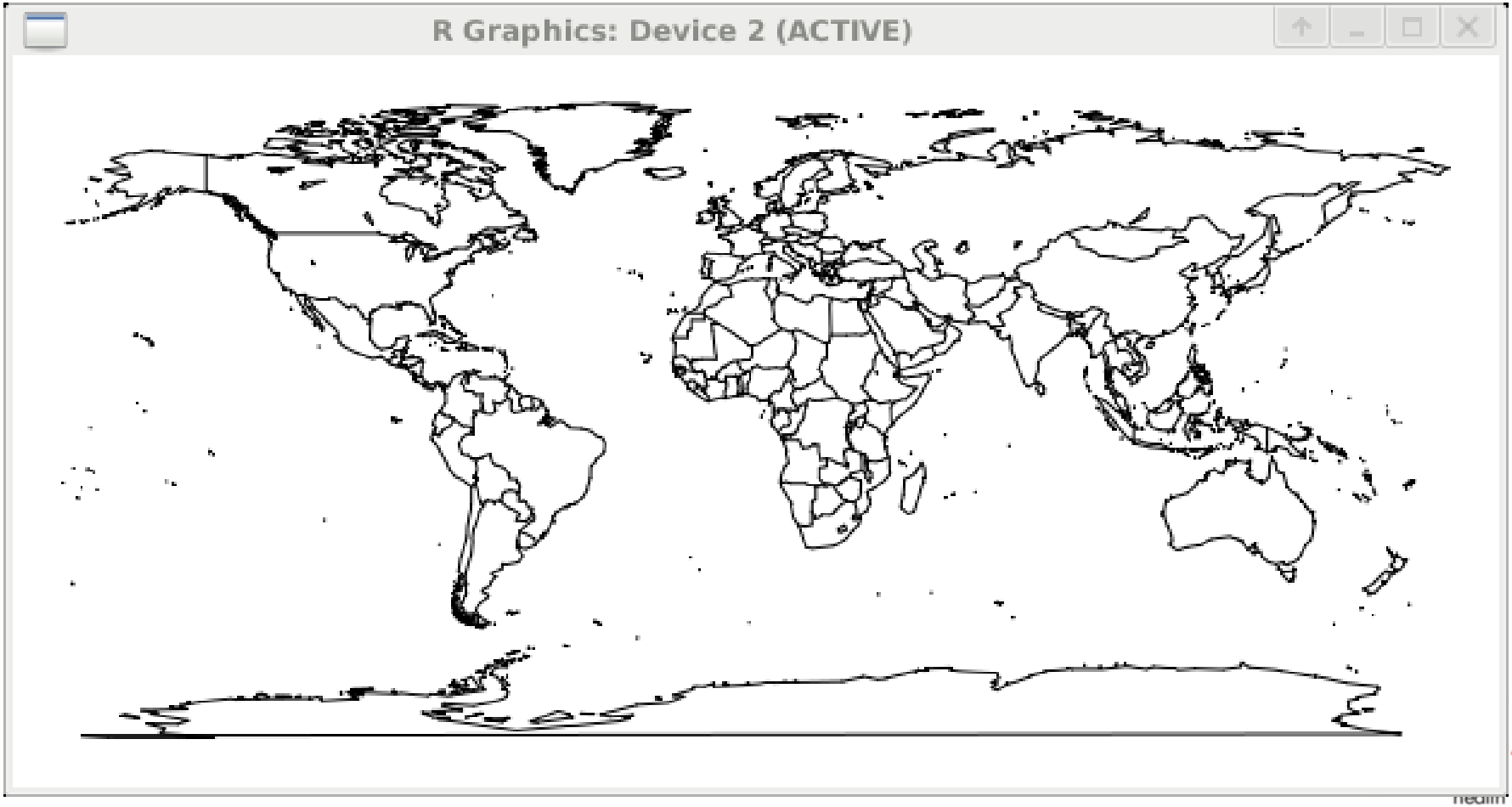
points, polygon, lines

- "Roll your own" maps
- `plot(xy,asp=1)`
- `polygon(xy,col=...)`
- `points(xy,pch=...)`
- `lines(xy,lwd=2,...)`

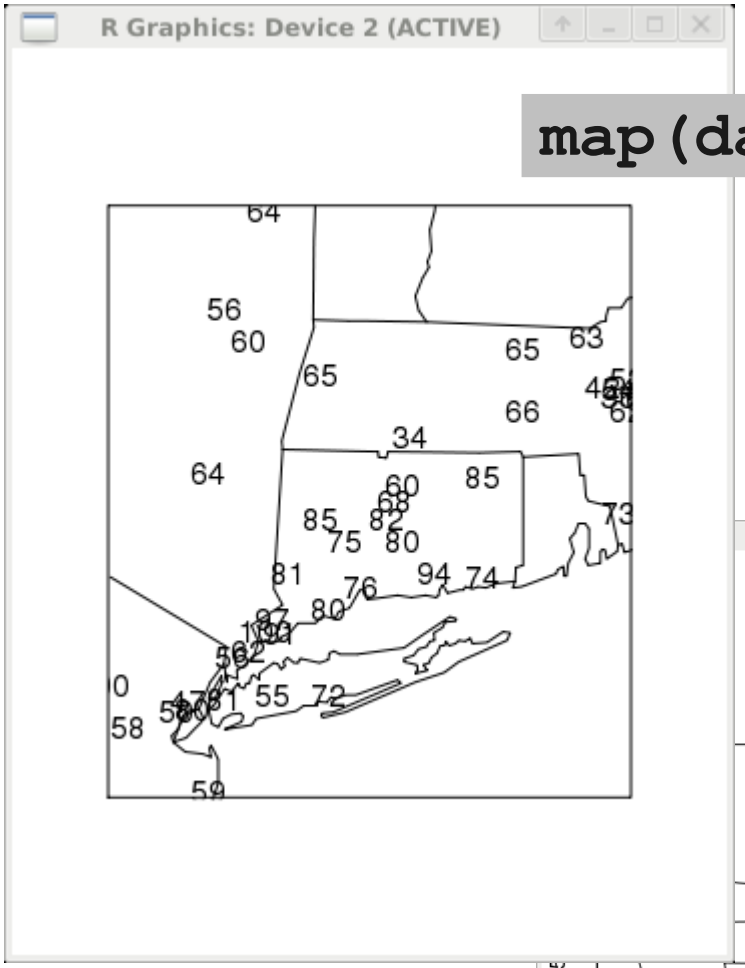


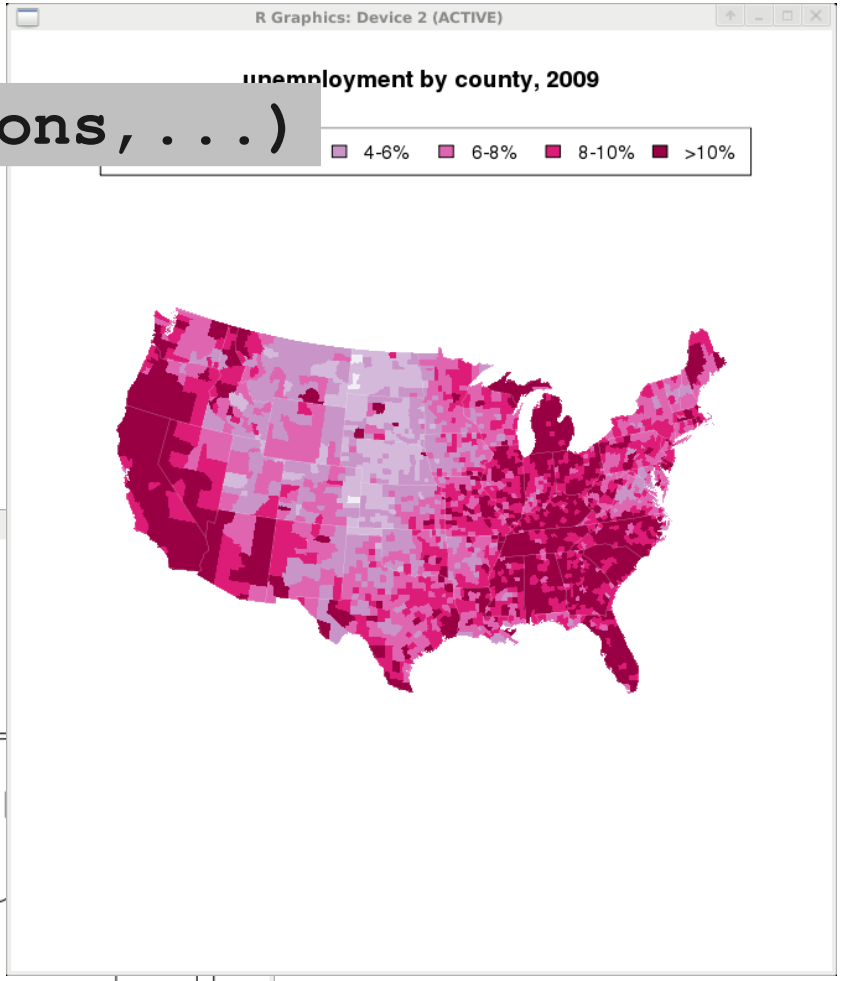
First Maps

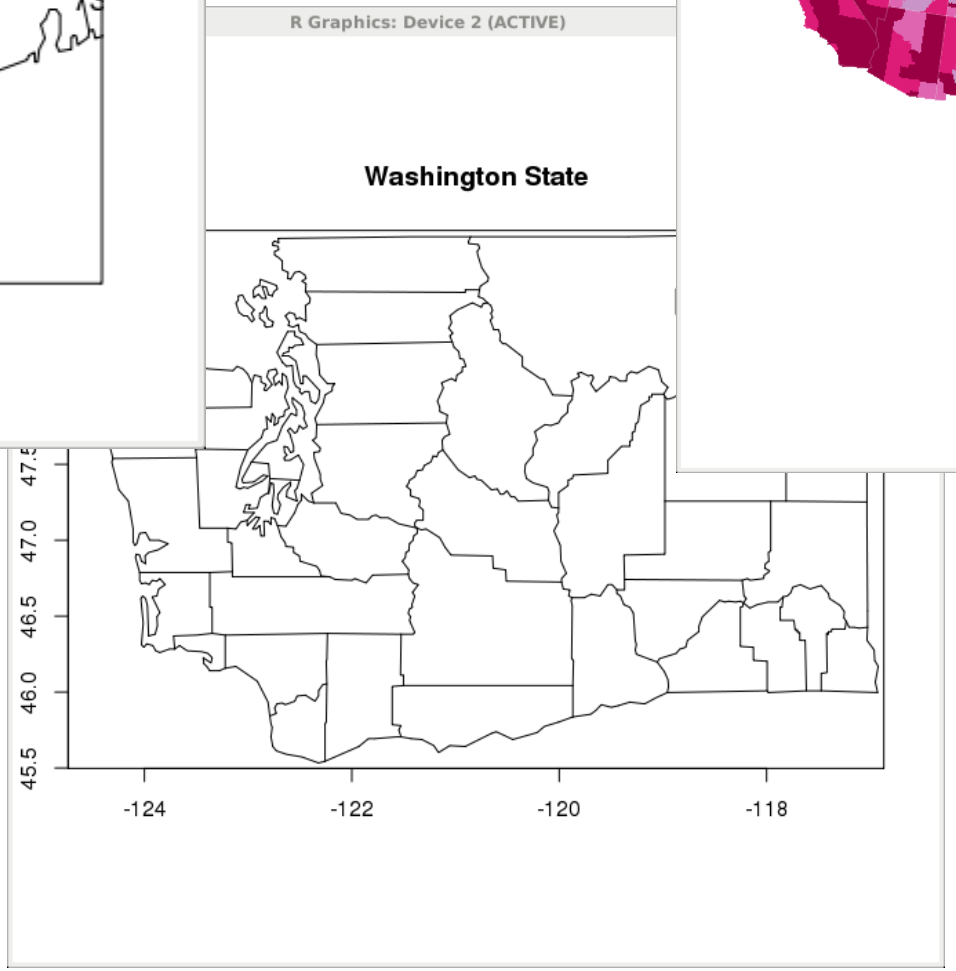
- The `map` package: Becker and Wilks
- **`library(maps);map()`**



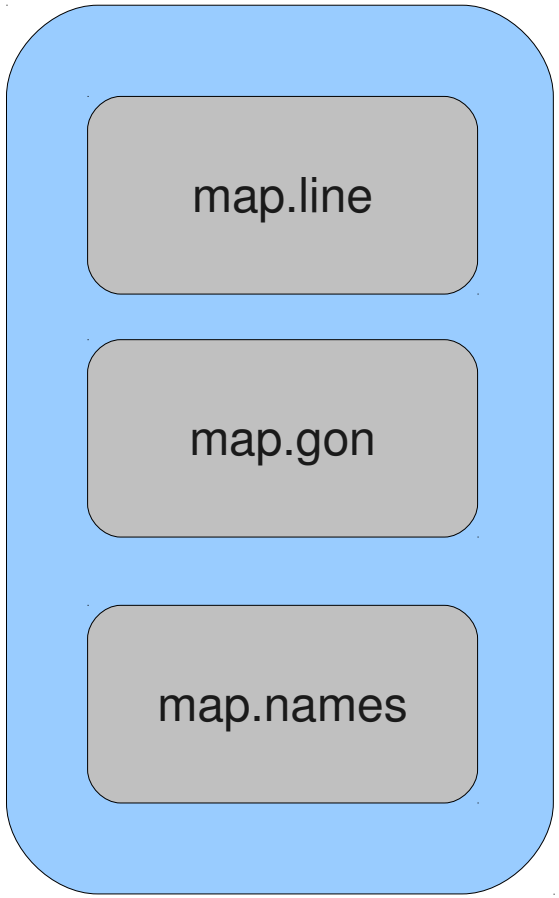
map(database,regions,...)





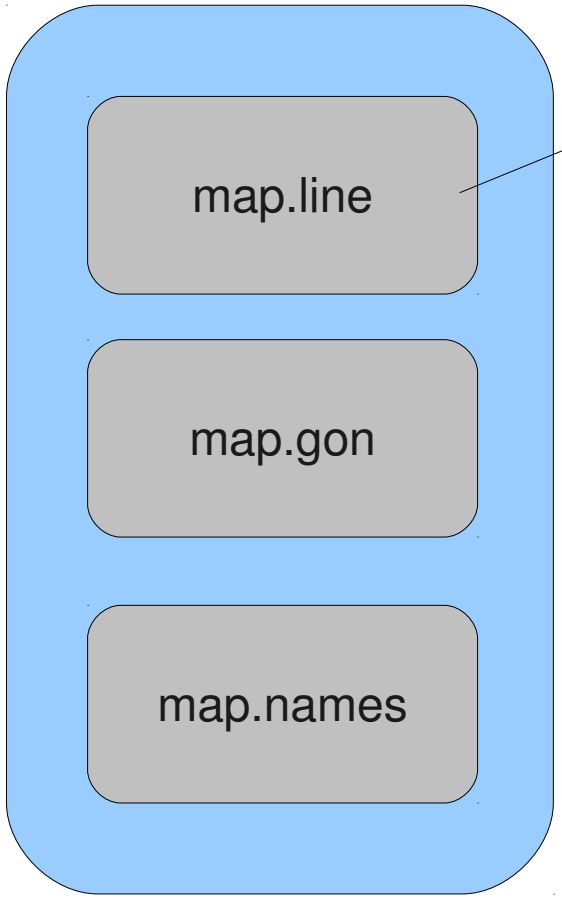


`map` database format



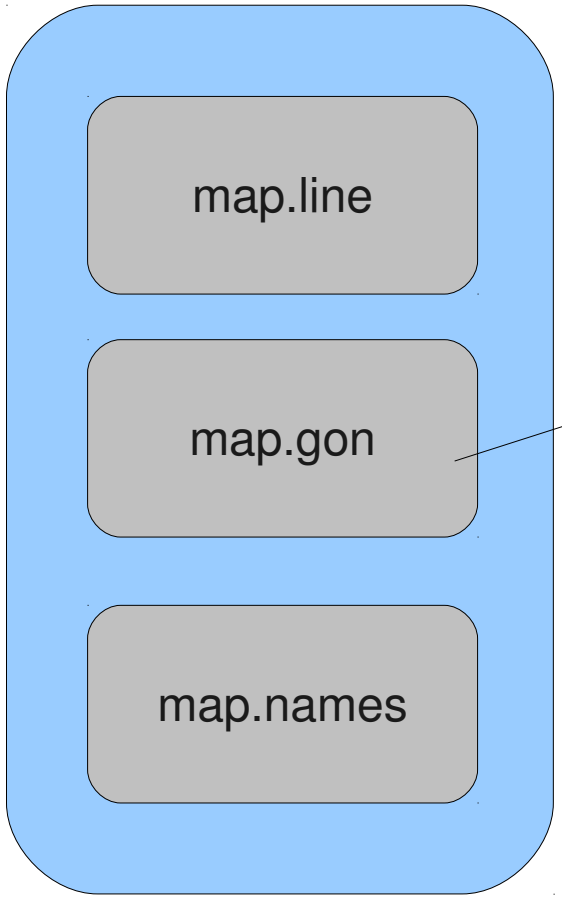
- To use your data with the maps package, first create these three files...

`map` database format



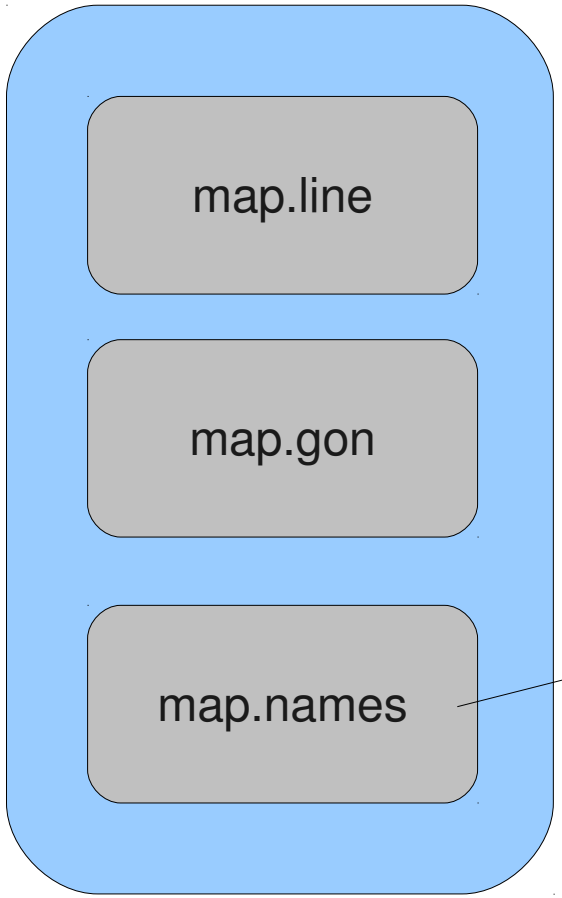

```
      2     1
4.07789 5.58567 4.04461 5.52577 3.91483
5.5291 3.86491 5.47585 3.79169 5.42926
3.78171 5.33941 3.80168 5.2828
3 3.86491 5.17634 3.88487 5.09314 3.99137
5.07317 4.13447 5.14306 4.17773 5.16303
4.28755 5.13973 4.35078 5.0
765 4.36077 5.12642
EOR
      3     1
4.36077 5.12642 4.37075 5.12642 4.47392
5.11643 4.5438 5.14306 4.71686 5.11643
4.77343 5.15304 4.77676 5.1996
3 4.69356 5.23624 4.61369 5.23624 4.57043
5.26619 4.51052 5.22626 4.51385 5.28616
4.49388 5.3627 4.46061 5.38
932 4.46726 5.38932
EOR
```

`map` data format



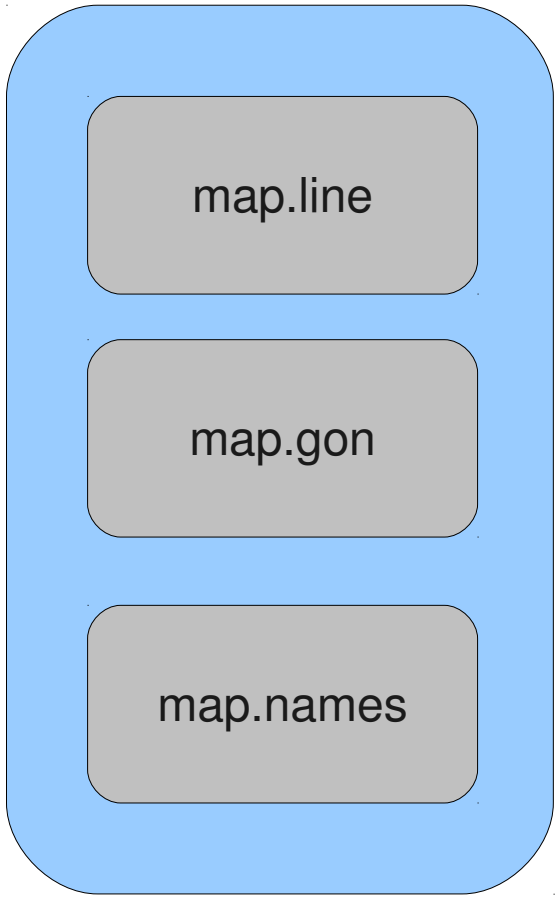

```
1  2  3  4
EOR
1  5  3  -6
EOR
2  -5
EOR
```

`map` data format

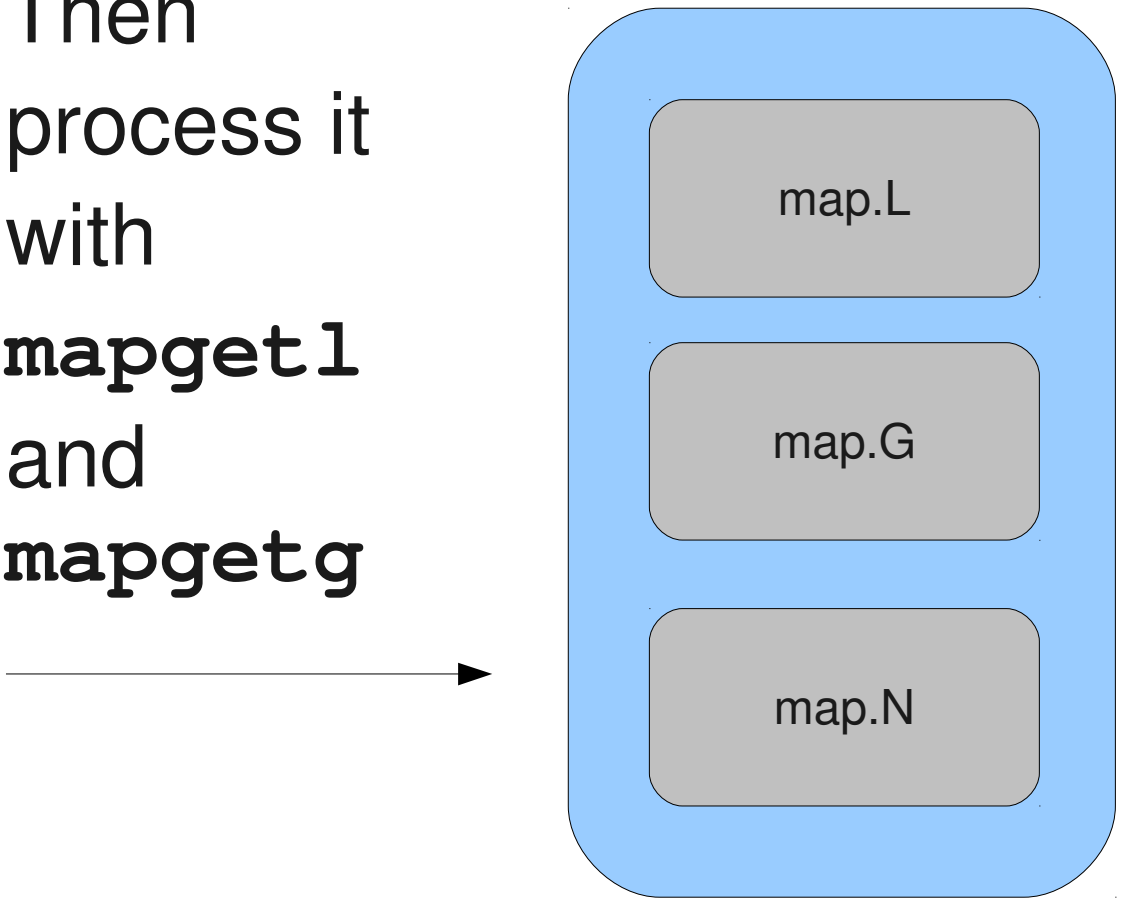



```
Scotland 1
Scotland 2
Scotland:Harris 3
Scotland:Harris 4
Scotland:Harris 5
Scotland:Harris 6
```

`map` data format



- Then process it with **`mapget1`** and **`mapgetg`**



`maps` package now maintained by Ray Brownrigg

CHICAS combining health information, computation and statistics

New Millennium

LANCASTER UNIVERSITY

- GIS spreading
- Many more map file formats (shapefile, 1998)
- Maps on the web (UMN Mapserver, mid-90s)
- Spatial Data Standards:

CHICAS
combining health information, computation and statistics

R didn't get left behind!

LANCASTER UNIVERSITY

- Perceived need for better maps
- Better spatial data handling
- Assorted little projects (my **`Rmap`**!)

CHICAS
combining health information, computation and statistics

Vector data - the sp classes

LANCASTER UNIVERSITY

- 2004 : A Geospatial Odyssey
- More than just polygons!
  - Classes for points, lines, grids of points and polygons.
- Geometry + Data for each feature
- Bivand, Pebesma, Gomez-Rubio
- “Simple Features” OGC spec

CHICAS
combining health information, computation and statistics

Create Points

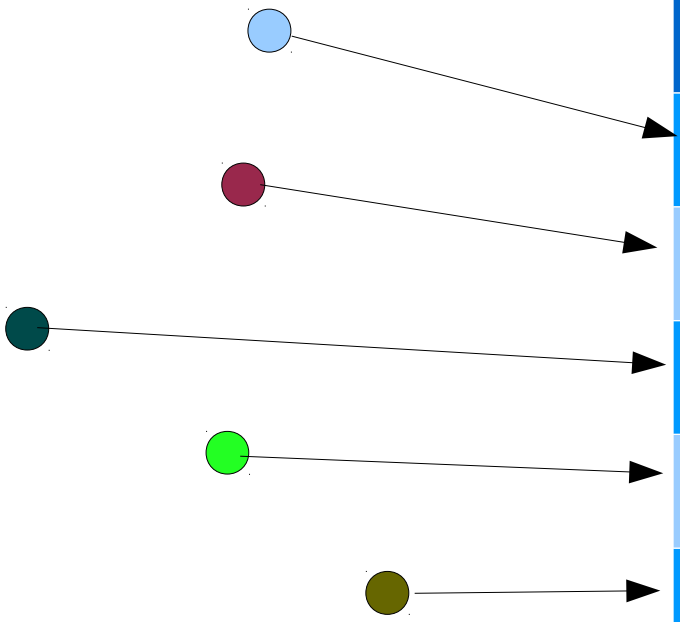
```
coords = cbind(x, y)

# create SpatialPoints
sp = SpatialPoints(coords)

# create a SpatialPointsDataFrame
spdf = SpatialPointsDataFrame(coords, data)
spdf = SpatialPointsDataFrame(sp, data)

# promote an ordinary data frame
coordinates(data) = cbind(x, y)
coordinates(data) = ~lon + lat

# and demote back to data frame
data = as.data.frame(data)
# or
data = data@data
```

LANCASTER UNIVERSITY

CHICAS combining health information, computation and statistics

Point Features

`SpatialPointsDataFrame`



ID	coordinate	Name	Pop
1	(x,y)	Alphaville	12000
2	(x,y)	Betaton	45255
3	(x,y)	Gamma CIty	5345
4	(x,y)	Delta VIllage	7745
5	(x,y)	Epsilondon	77514

Manipulation

```
BigTowns = Towns[Towns$pop>10000,]
SmallTowns = Towns[Towns$pop<5000,]

BigTowns = subset(Towns,pop>10000)
BigTowns = subset(Towns,Towns$pop>10000)

BigSmall = rbind(BigTowns,SmallTowns)


xy = coordinates(Towns)

TownData = Towns@data

TownData = as.data.frame(Towns)
```

Treat like a data frame...

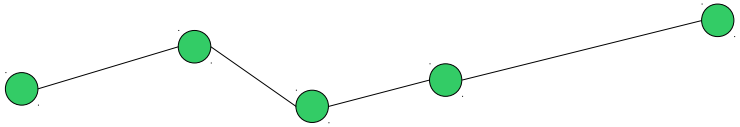
...almost (this fails)

this works

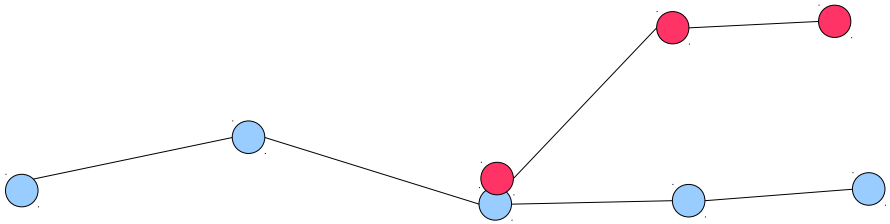
Retrieve coords and data

Create Line Objects

A **`Line`** is a single chain of points



```
L1 = Line(cbind(x1,y1))
```

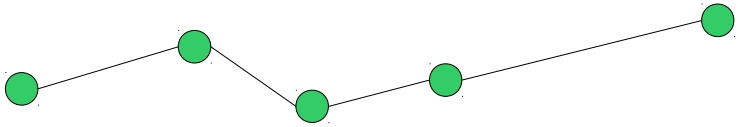


```
L2 = Line(cbind(x2,y2))
```

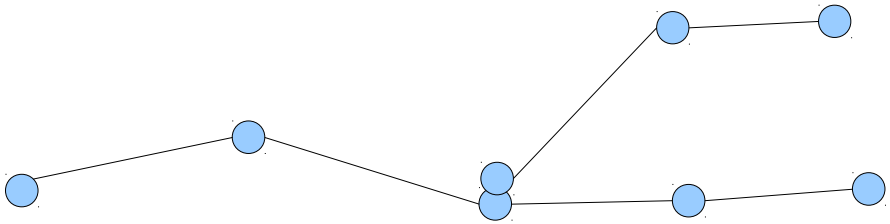
```
L3 = Line(cbind(x3,y3))
```

Create Lines Objects

A **`Lines`** is a list of chains with an ID



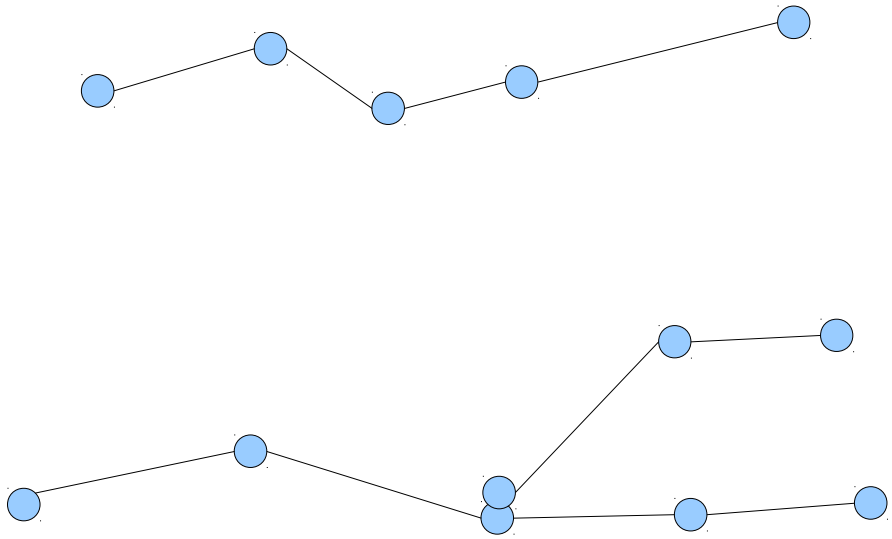
```
Ls1 = Lines(list(L1),ID="a")
```



```
Ls2 = Lines(list(L2,L3),ID="b")
```

Create SpatialLines Object

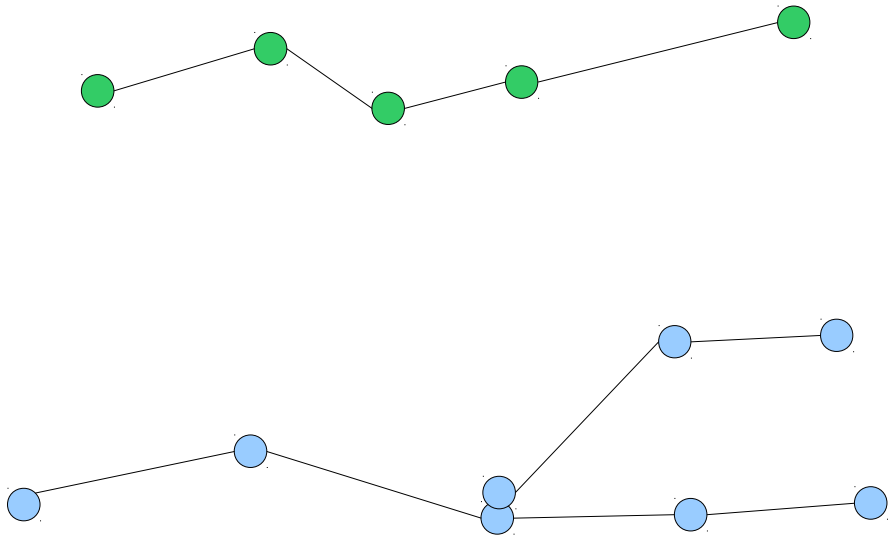
A **`SpatialLines`** is a list of **`Lines`**



```
SL12 = SpatialLines(list(Ls1,Ls2))
```

Create SpatialLinesDataFrame

A **SpatialLinesDataFrame** is a **SpatialLines** with a matching **Data Frame**

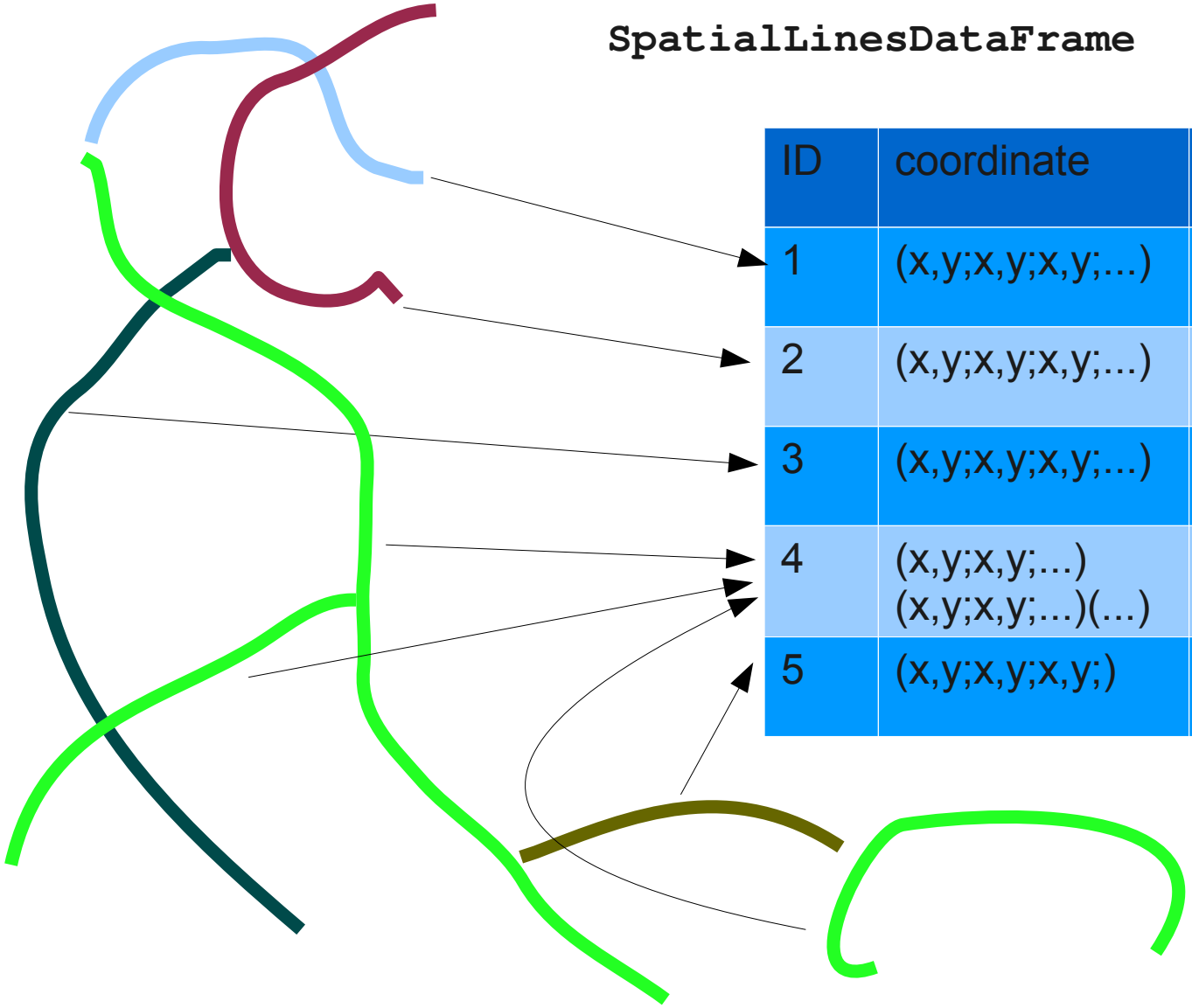


```
SLDF = SpatialLinesDataFrame(
           SL12,
           data.frame(
               Z=c("road","river"),
               row.names=c("a","b")
           )
       )
```

Matching on the **row.names** to the **ID** values or optionally by order

CHICAS combining health information, computation and statistics

Line Features



SpatialLinesDataFrame

ID	coordinate	Name	Quality
1	(x,y;x,y;x,y;...)	River Alpha	Good
2	(x,y;x,y;x,y;...)	Bravo Beck	Good
3	(x,y;x,y;x,y;...)	Charlie River	Fair
4	(x,y;x,y;...) (x,y;x,y;...)(...)	Delta Drains	Poor
5	(x,y;x,y;x,y;)	Echo Stream	Good

LANCASTER UNIVERSITY

CHICAS
combining health information, computation and statistics

Polygons

LANCASTER UNIVERSITY

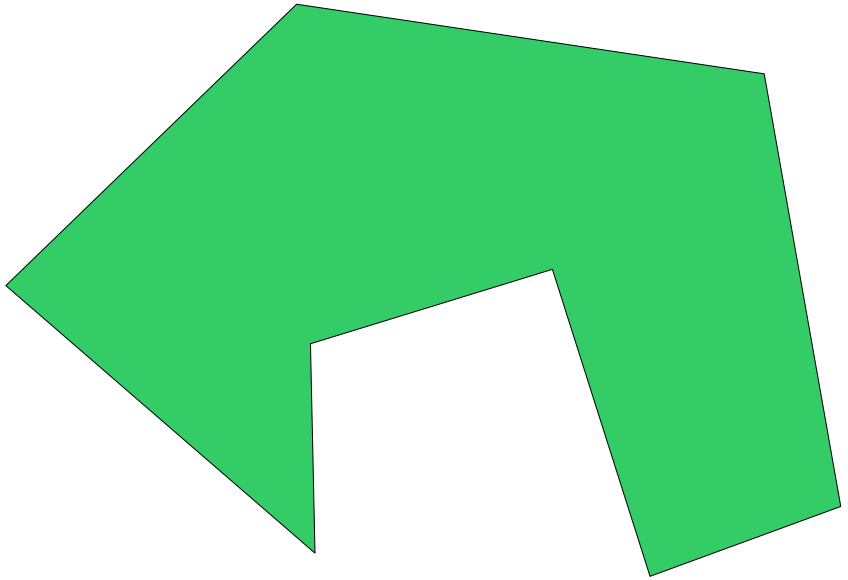
Now it starts to get complicated

CHICAS
combining
health information,
computation and statistics

Single Ring Feature

Make sure your coordinates connect:

```
c1 = cbind(x1,y1)
r1 = rbind(c1,c1[1,])
```



A **`Polygon`** is a single ring

```
P1 = Polygon(r1)
```

A **`Polygons`** is a list of **`Polygon`** objects with an **`ID`**

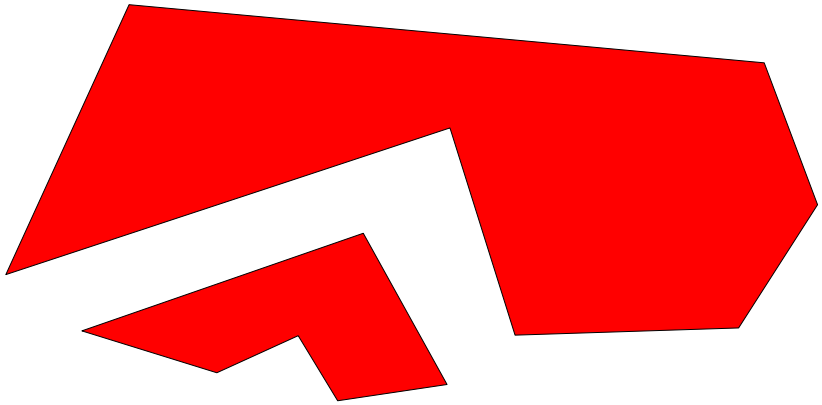
```
Ps1 = Polygons(list(P1,ID="a"))
```

CHICAS
combining health information, computation and statistics

Double Ring Feature

Make sure your coordinates connect:

```
c2a = cbind(x2a,y2a)
r2a = rbind(c2a,c2a[1,])
c2b = cbind(x2b,y2b)
r2b = rbind(c2b,c2b[1,])
```



Make two single rings:

```
P2a = Polygon(r2a)
P2b = Polygon(r2b)
```

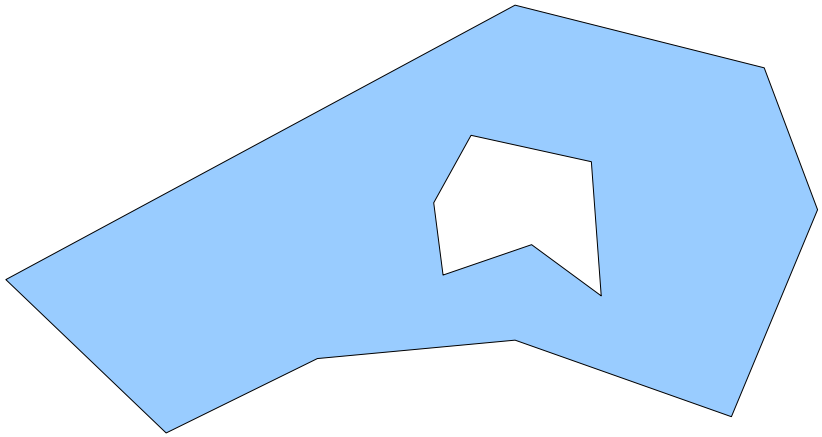
Make a Polygons object with two rings

```
Ps2 = Polygons(list(P2a,P2b),ID="b"))
```

CHICAS combining health information, computation and statistics

Feature with hole

```
hc = cbind(xh,yh)
rh = rbind(hc,hc[1,])
```



Make a holey **`Polygon`**:

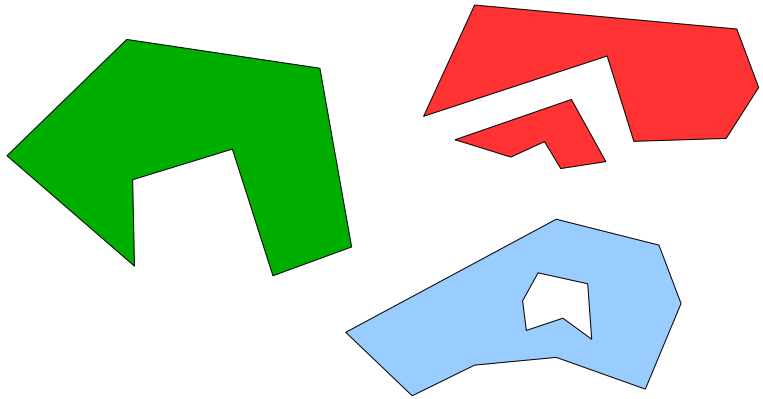
```
H1 = Polygon(rh, hole=TRUE)
```

Make a **`Polygons`** object with two rings, one the hole

```
Ps3 = Polygons(list(P3,H1),ID="c"))
```

SpatialPolygonsDataFrame

```
SPs = SpatialPolygons(
    list(Ps1,Ps2,Ps3)
)
```



```
SPDF = SpatialPolygonsDataFrame(
    SPs,
    data.frame(
      info=c("single","double","hole"),
      row.names=c("a","b","c")
      )
    )
```

And back again

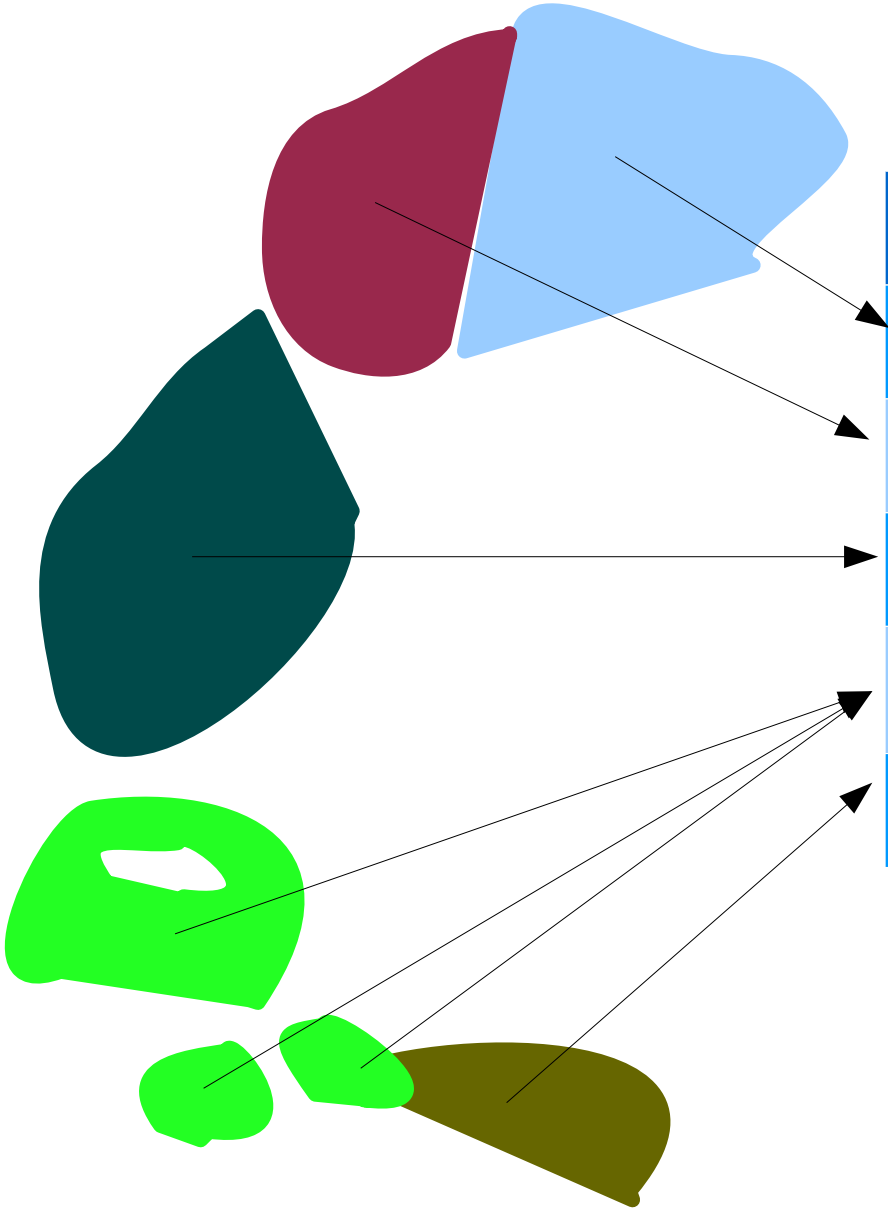
LANCASTER UNIVERSITY

For ring `j` of feature `i`:

```
> SPDF@polygons[[i]]@Polygons[[j]]@coords
              x        y
 [1,] 42.53623 76.85066
 [2,] 42.57308 76.79820
 [3,] 43.05132 74.56176
...
```

CHICAS
combining health information, computation and statistics

Polygon Features



SpatialPolygonsDataFrame

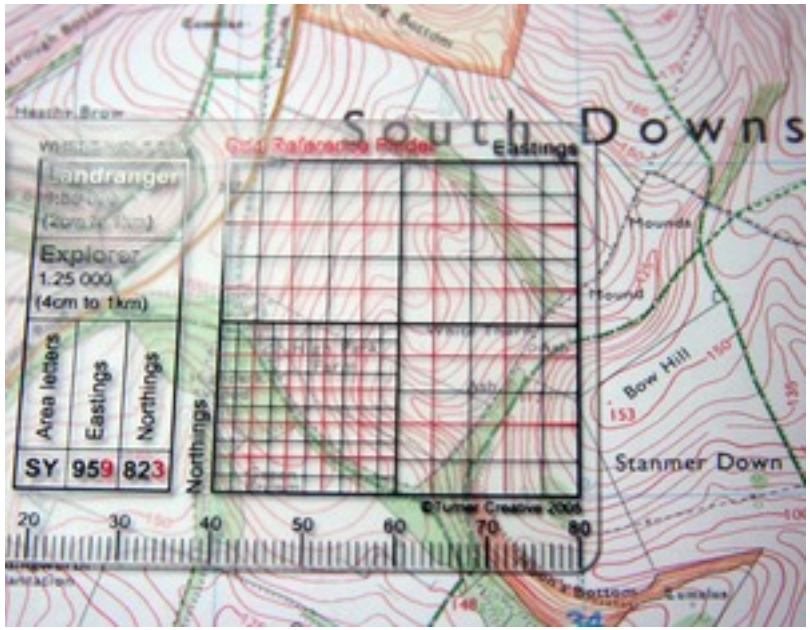
ID	coordinates	Name	Pop.
1	(x,y;x,y;x,y;...)	Alphashire	12365
2	(x,y;x,y;x,y;...)	Bravoshire	45366
3	(x,y;x,y;x,y;...)	County Charlie	78725
4	(x,y;x,y;...) (x,y;x,y;...)(...)	Delta District	77439
5	(x,y;x,y;x,y;)	Echoville	34545

LANCASTER UNIVERSITY

CHICAS
combining health information, computation and statistics

Coordinate Systems

- Because the Earth isn't flat
- But small parts of it are near enough
- So lots of coordinate systems exist
- Anything `Spatial*` can have one



CHICAS
combining health information, computation and statistics

CRS and proj4string

without:

```
sp = SpatialPoints(coords)
```

at construction time:

```
sp = SpatialPoints(
          coords,
          proj4string = CRS("+init=epsg:4326")
       )
```

After the event:

```
proj4string(sp) = CRS("+init=epsg:4326")
```

And for Spatial*DataFrames

LANCASTER UNIVERSITY

```
spdf = SpatialPointsDataFrame(
    sp,
    data,
    proj4string=CRS("+init=epsg:4326")
    )

proj4string(spdf)=CRS("+init=epsg:4326")
```

CHICAS
combining health information, computation and statistics

What are these proj4strings?

LANCASTER UNIVERSITY

- International standard codes for coordinate reference systems
- **`+init=epsg:4326`** lat-long, GPS
- **`+init=epsg:27700`** UK Grid
- ~~**`+init=epsg:900913`**~~ “google”
- **`+init=epsg:3857`** Spherical Mercator
- **`+proj=utm +zone=19 +north`**

Search **`spatialreference.org`** for UTM zones

CHICAS
combining health information, computation and statistics

Transform Spatial* Objects

LANCASTER UNIVERSITY

- Assigning a CRS doesn't change the numbers
- So assigning the wrong CRS really messes things up
- Convert from one to another with **`spTransform`**

  If **`sp`** is **`SpatialPoints`** from my GPS:

```
proj4string(sp)=CRS("+init=epsg:4326")
```

  And to put on a UK National Grid map:

```
spUK = spTransform(sp,CRS("+init=epsg:27700"))
```

- The coordinates are now in metres!

CHICAS combining health information, computation and statistics

Raster Data – regular grids

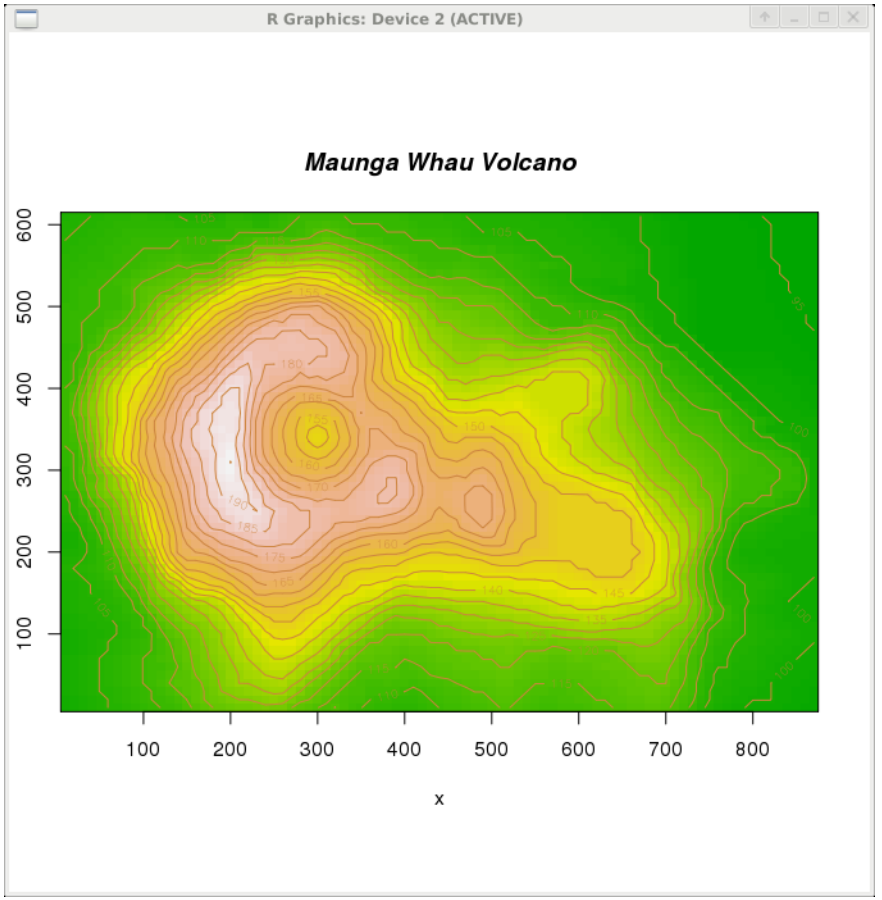
- Satellite Imagery
  - Photography
  - Radar Height
  - Other Sensors
- Count Data
  - Animals in quadrats
- Modelled Values
  - Climate Data
  - Interpolations

LANCASTER UNIVERSITY

CHICAS
combining health information, computation and statistics

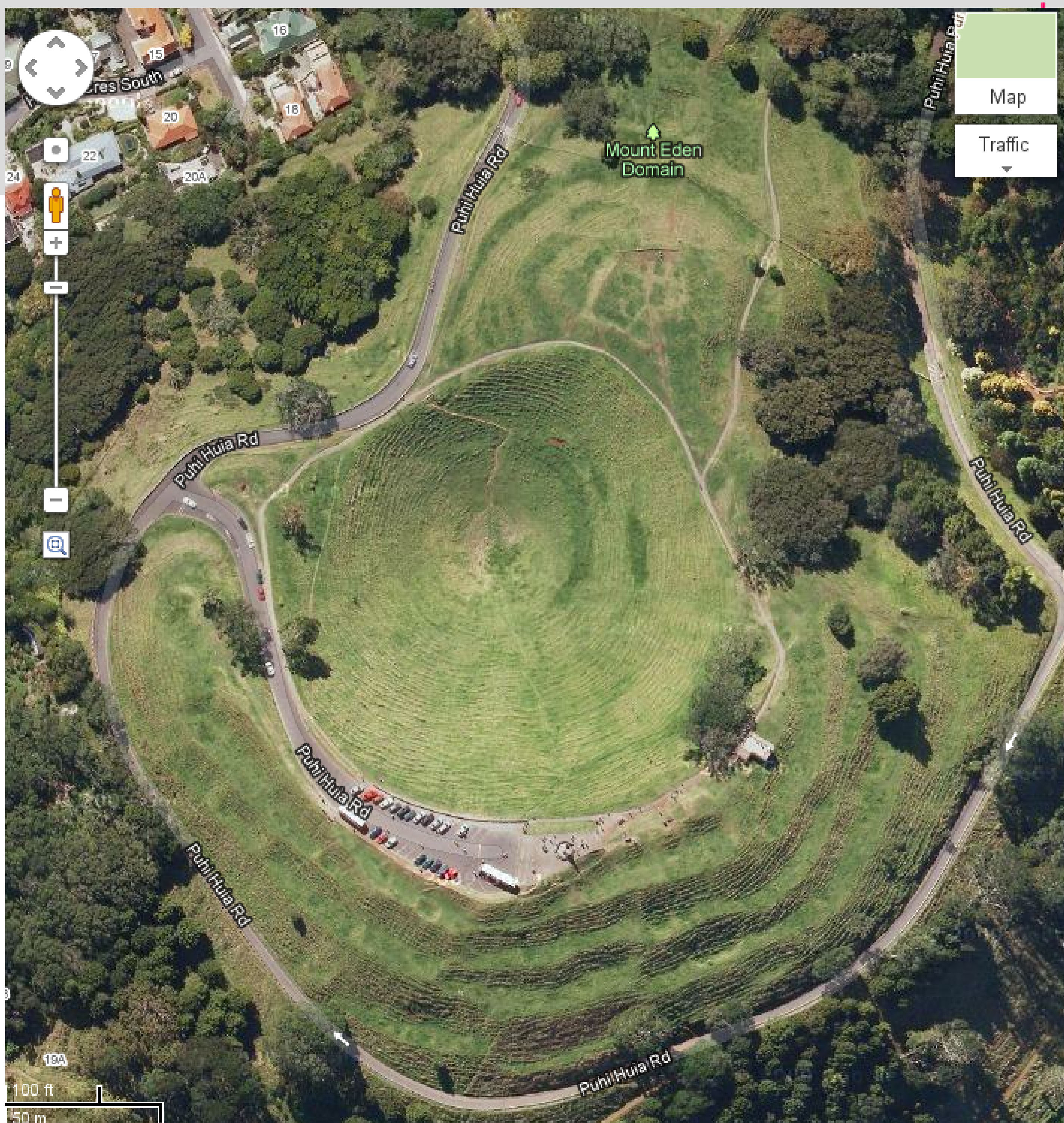
Rasters in R - history

- The **`image`** function
- Works on a matrix
- X and Y vectors define position



CHICAS
combining health information, computation and statistics

Volcano?



computation and statistics

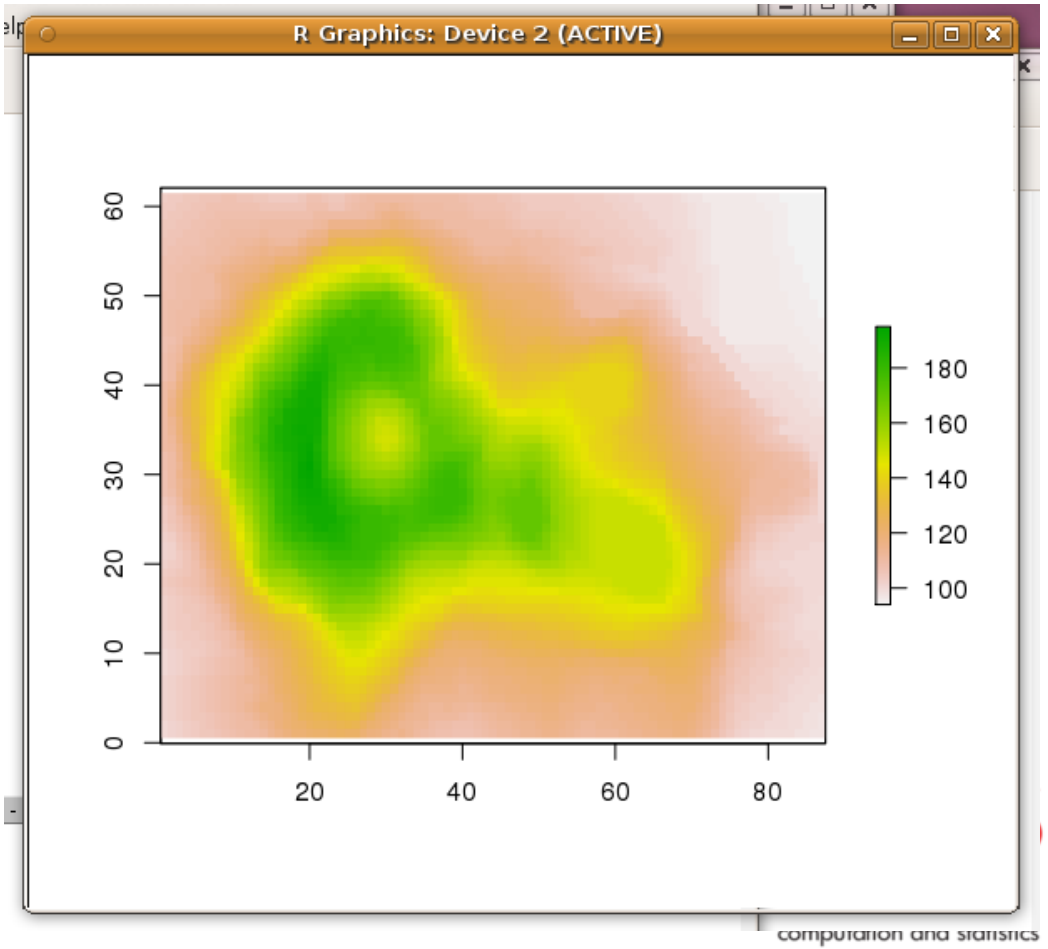
Volcano


5/5/2012 12 am
© 2012 Cnes/Spot Image
© 2012 Whereis® Sensis Pty Ltd
68 m
Google

The `raster` package

- Create from vector x, vector y, matrix z

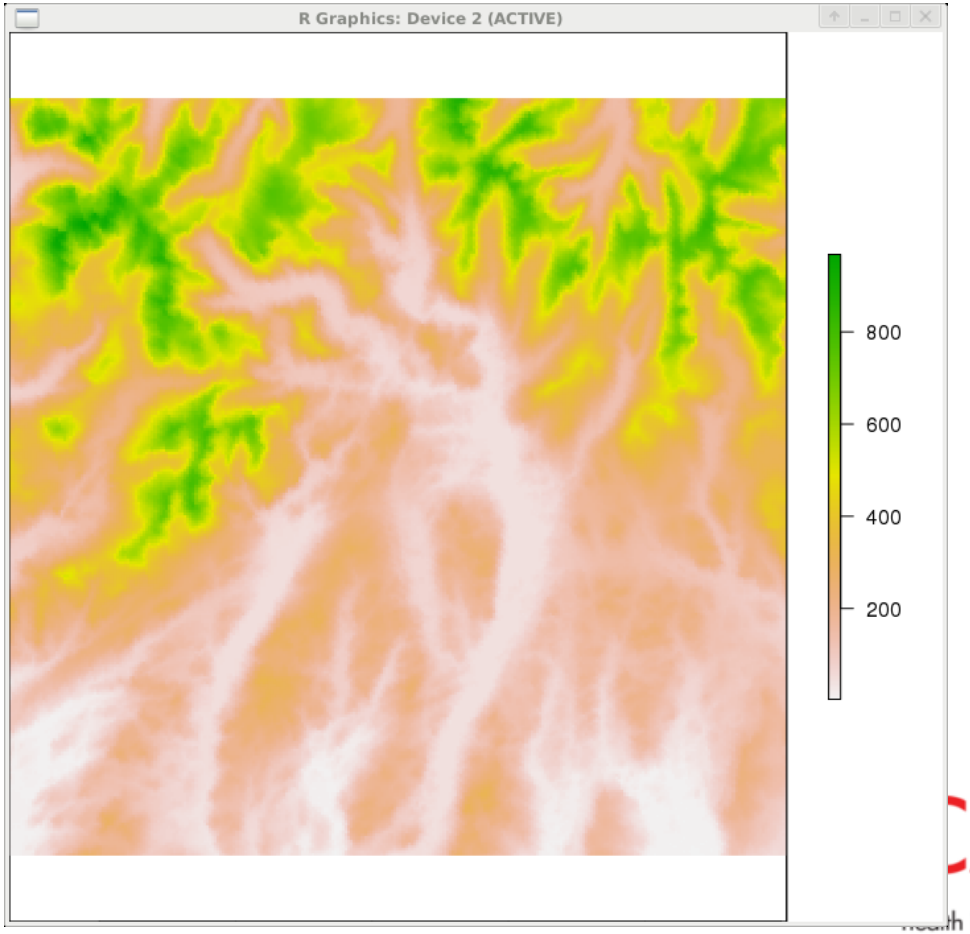
```
r = raster(list(x=x,y=y,z=z))
plot(r)
```



The `raster` package

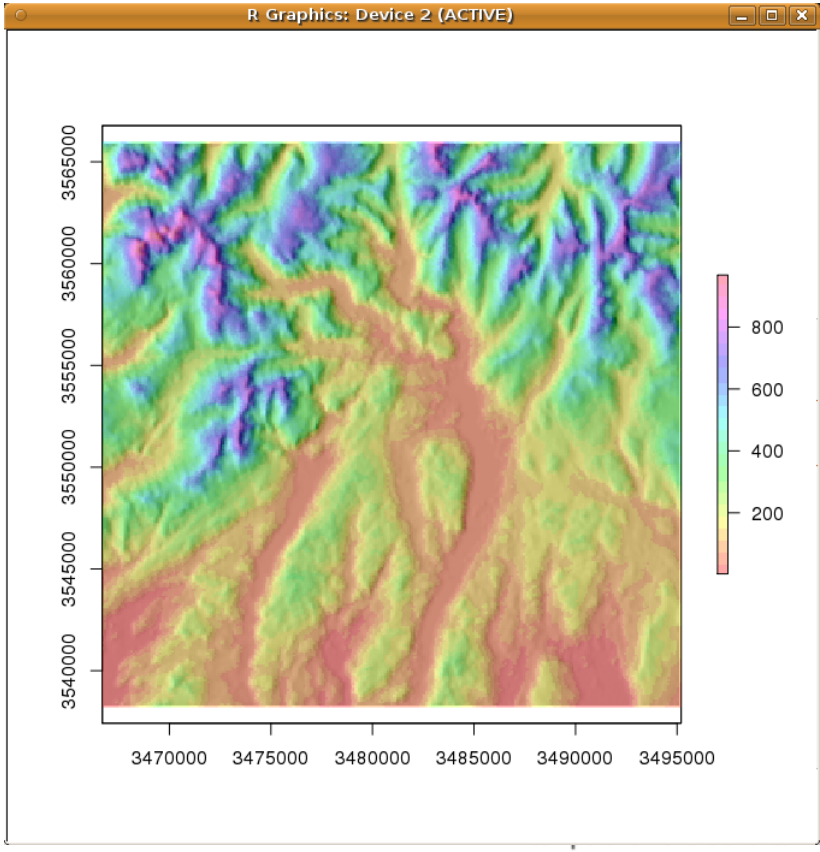
- Read and plot standard raster files

```
dem = raster('cumbriaDem.tif')
plot(dem)
```



Raster ops

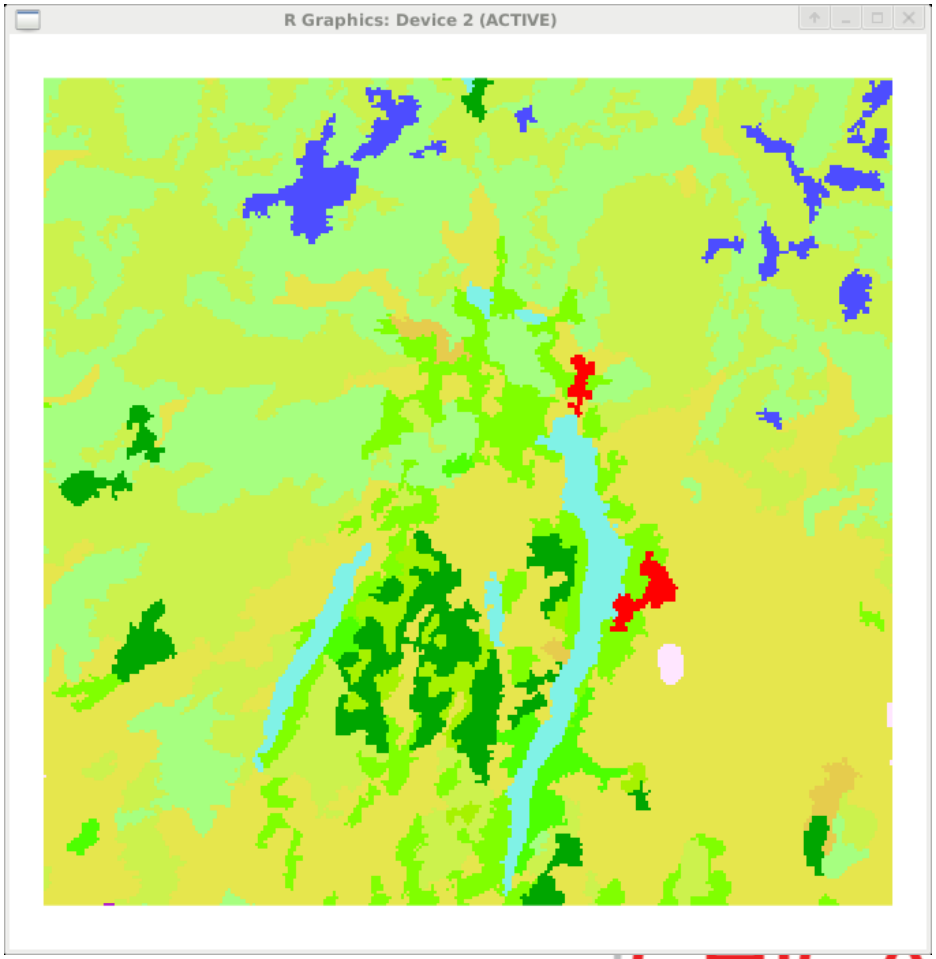
```
highGround = dem > 600
slope = terrain(dem,opt='slope')
aspect = terrain(dem,opt='aspect')
hill = hillShade(slope,aspect,40,270)
plot(hill,col=grey(0:100/100)
      ,legend=FALSE)
plot(dem,
  col=rainbow(25,alpha=0.35),
  add=TRUE)
```



Categorical rasters

- `use = raster('cumbria.tif');plot(use)`

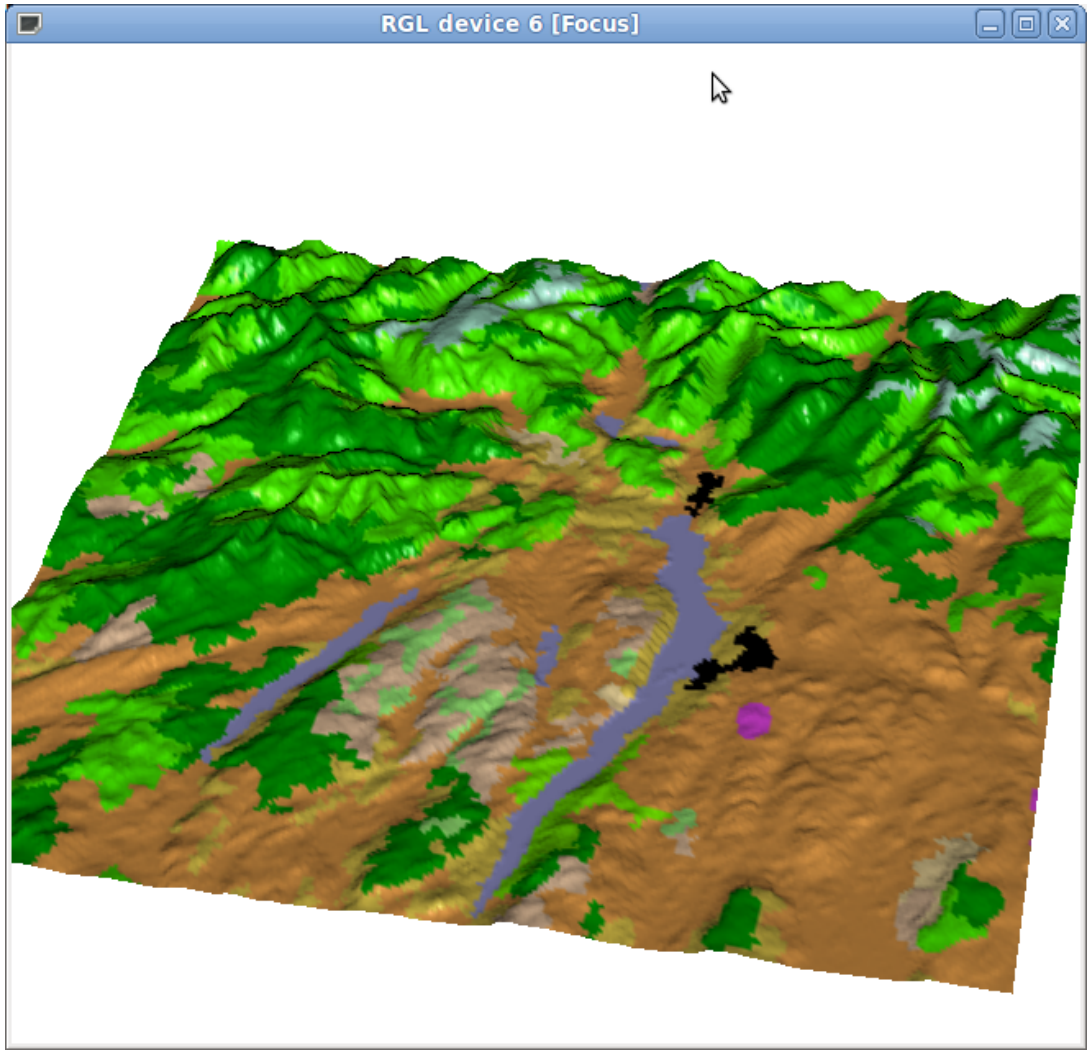
```
> as.matrix(table(values(use)))
     [,1]
2     325
7       4
11    121
18  24399
21    425
23   5087
24   3084
[etc]
```



3d visuals – `rasterVis`

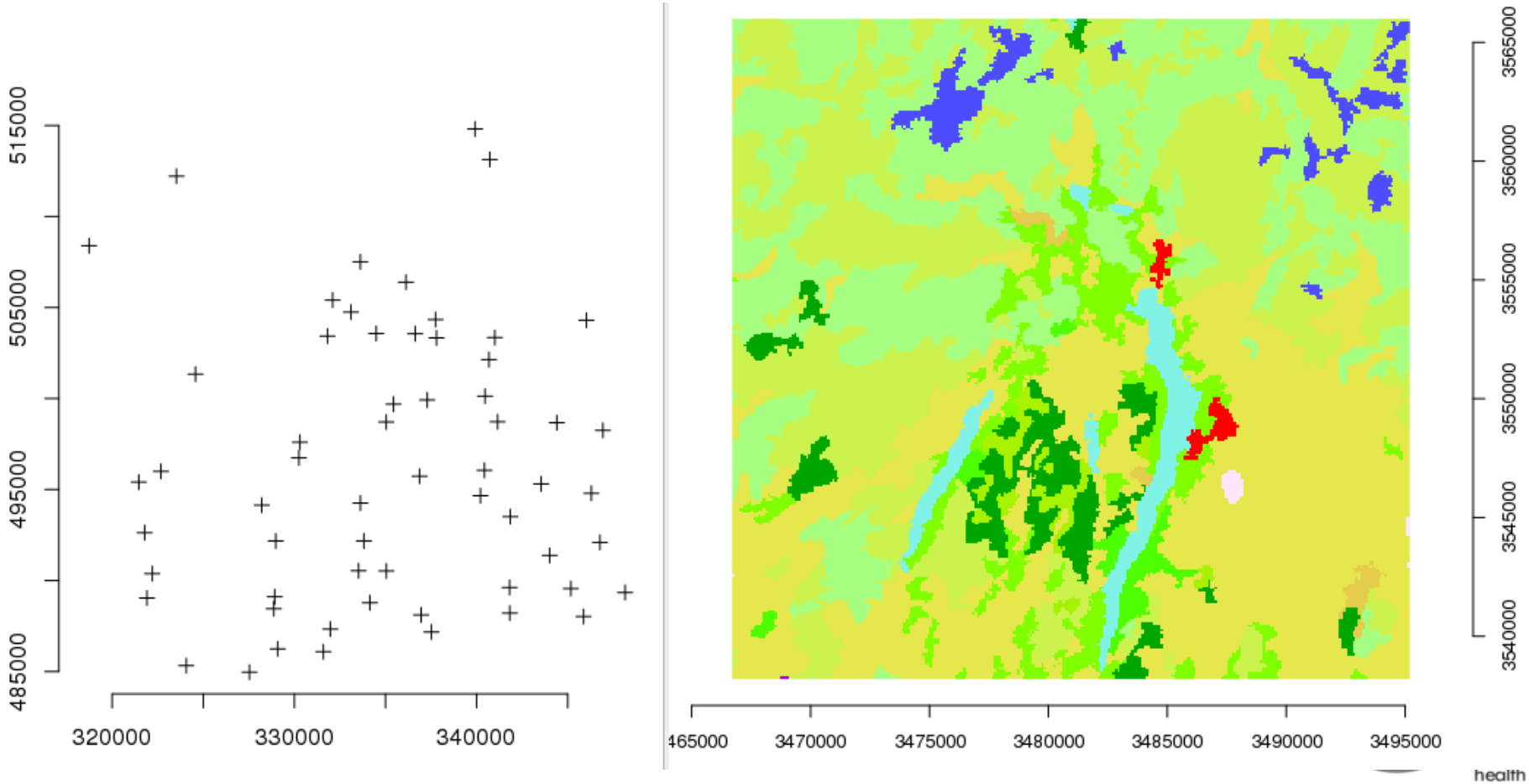
- Drape land-use over dem

```
plot3D(dem,
       drape=use)
```



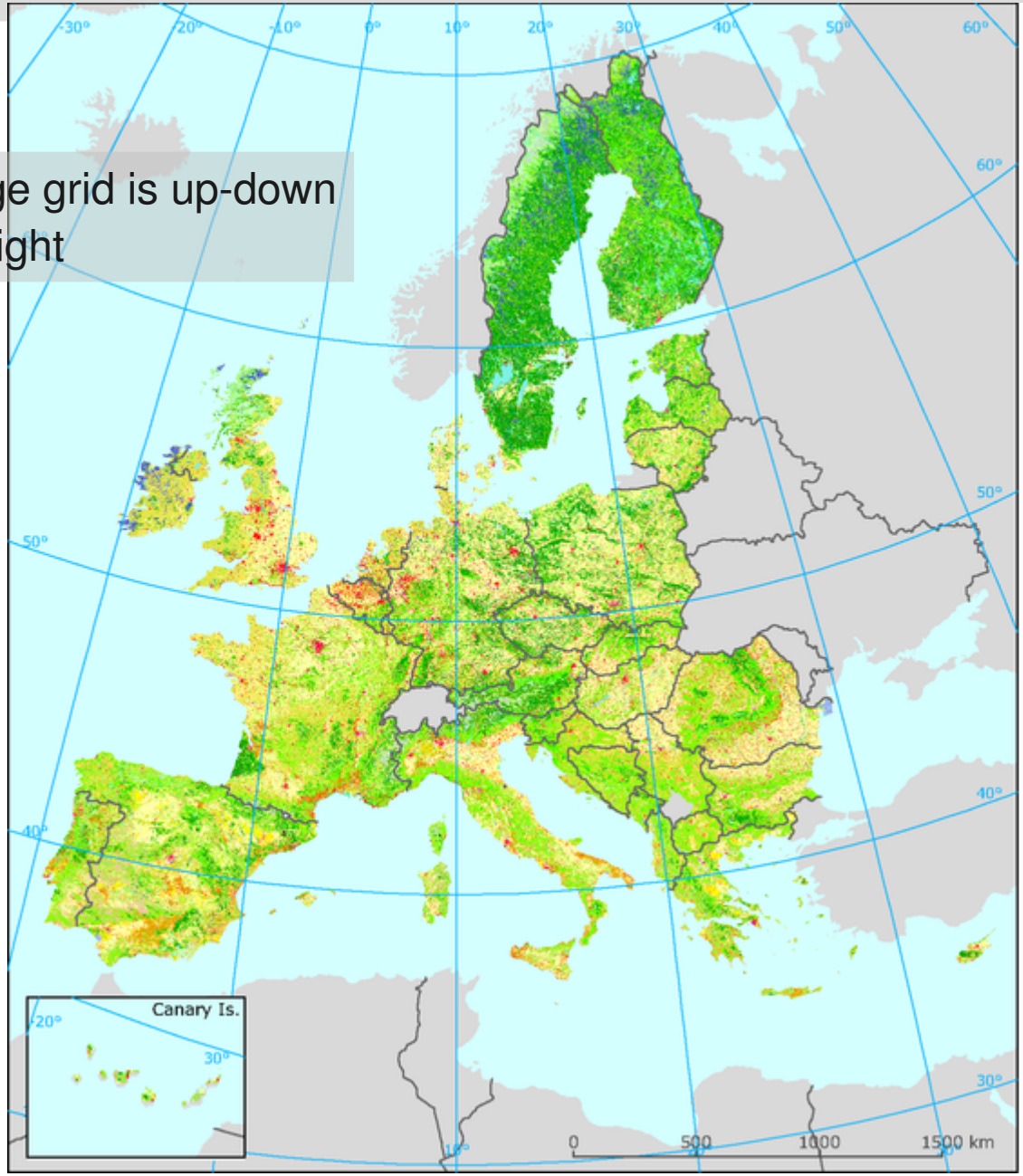
Projections and CRS

```
> use = raster('/data/CorineLandCover/cumbria.tif')
> plot(settlements)
> plot(use)
```



Projection

Image grid is up-down
left right



LANCASTER UNIVERSITY

CAS
combining
health information,
computation and statistics

Projections

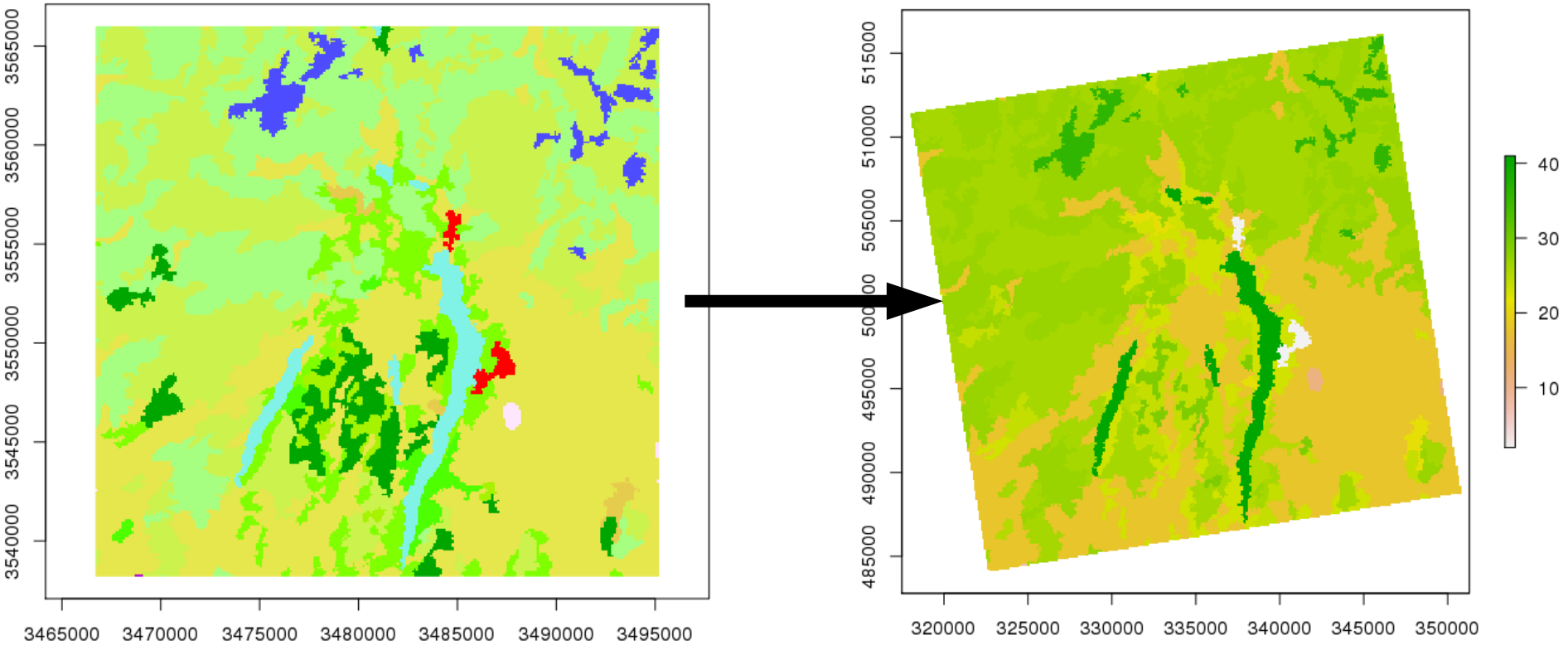
- rasters have a CRS

```
> use = raster('/data/CorineLandCover/cumbria.tif')
> use
class       : RasterLayer
dimensions  : 278, 285, 79230  (nrow, ncol, ncell)
resolution  : 100, 100  (x, y)
extent      : 3466700, 3495200, 3538200, 3566000
coord. ref. : +proj=laea +lat_0=52 +lon_0=10
              +x_0=4321000 +y_0=3210000 +ellps=GRS80
              +units=m +no_defs
values      : /data/CorineLandCover/cumbria.tif
min value   : 2
max value   : 41
> projection(use)
 [1] "+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000
+y_0=3210000 +ellps=GRS80 +units=m +no_defs"
```

Reproject Raster

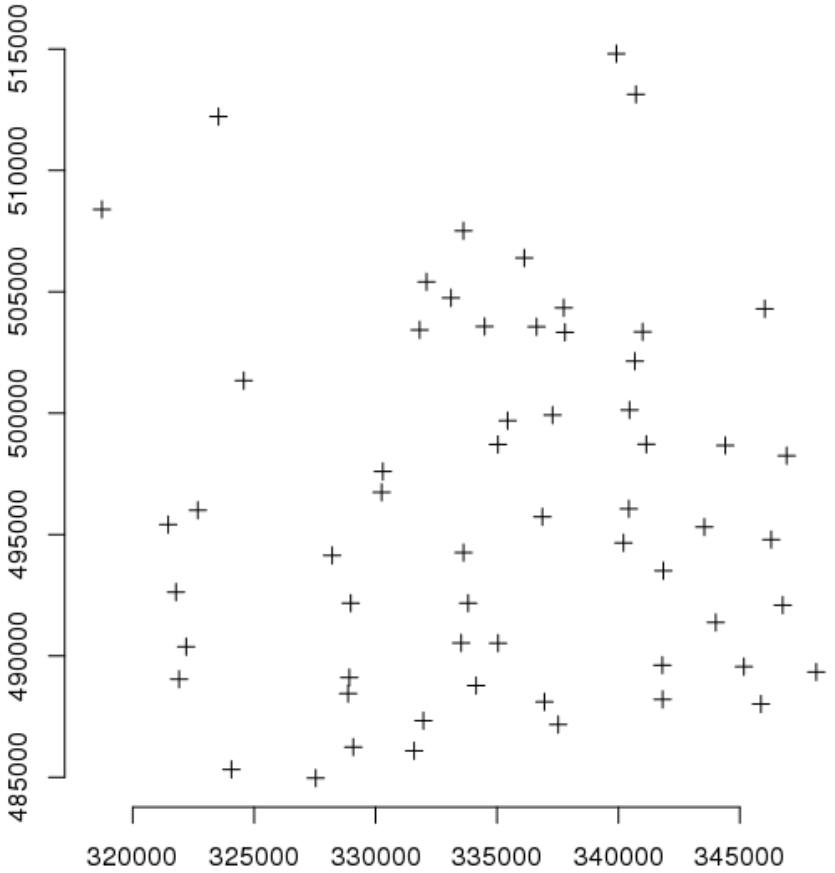
- Involves a 'warp'

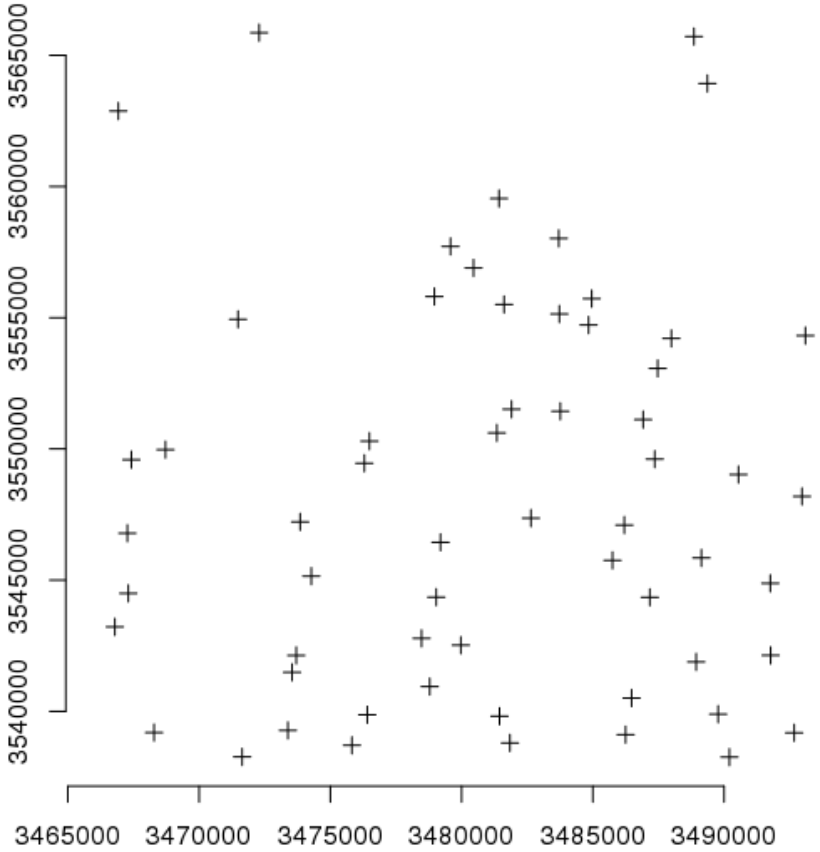
```
> useOS = projectRaster(use,res=100,
    crs="+init=epsg:27700",method="ngb")
```



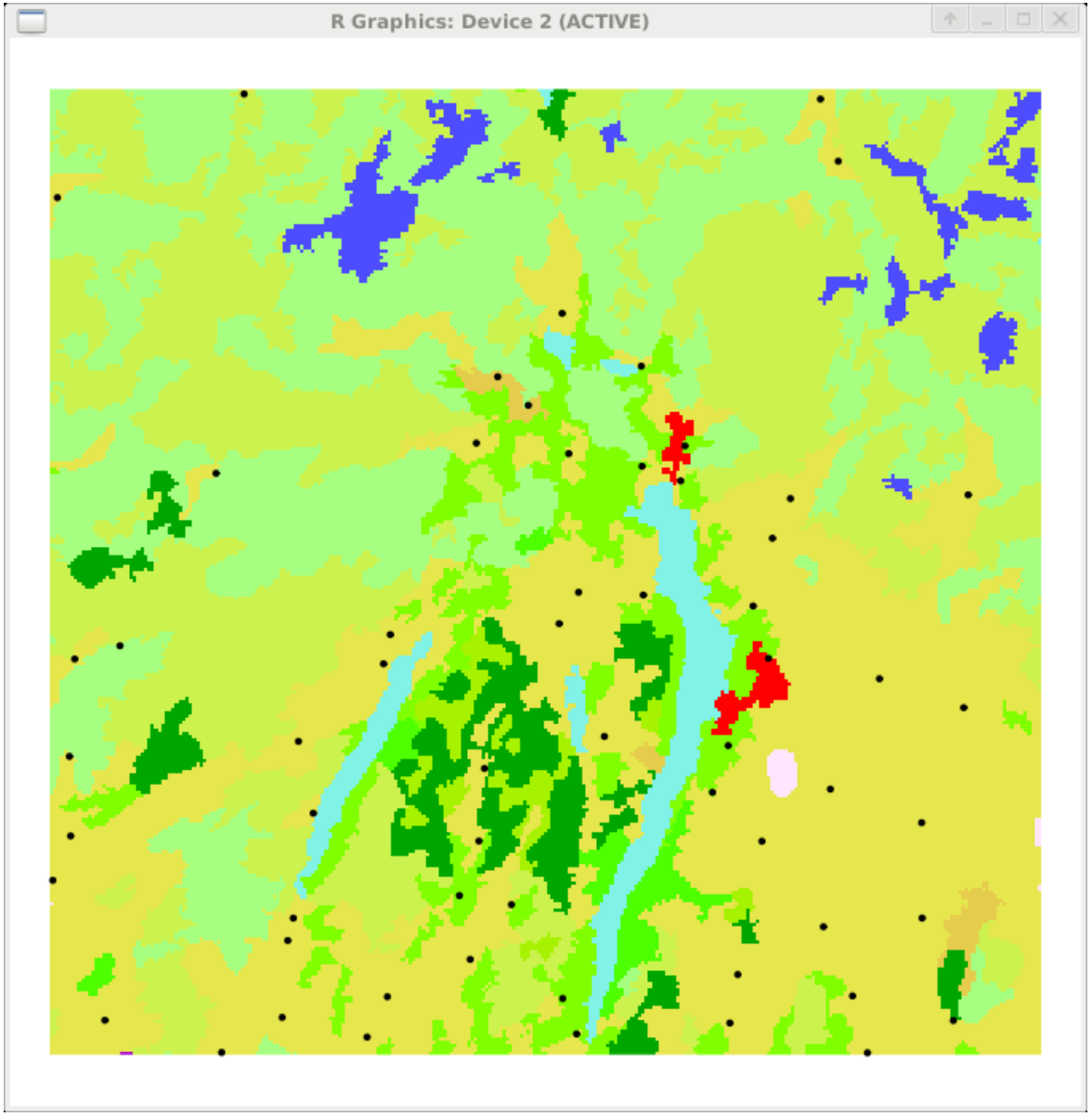
Better to transform points

```
> settlementsP = spTransform(settlements,
                             CRS(projection(use)))
```





Points and Raster



Now lets go to work

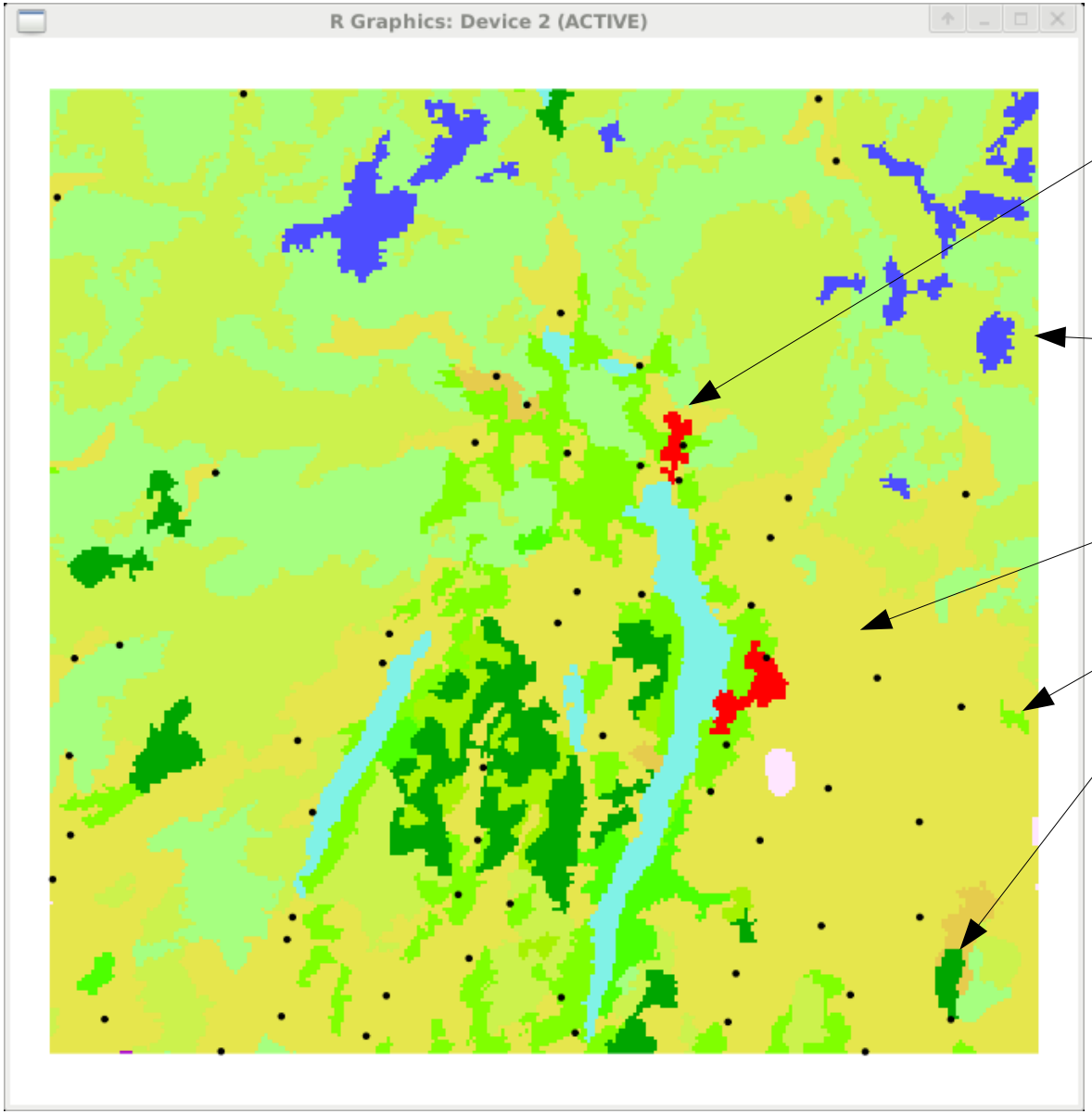
- Data created
- Coordinates corrected

- Lets do some analysis!

LANCASTER UNIVERSITY

CHICAS
combining health information, computation and statistics

Settlements and Land Cover



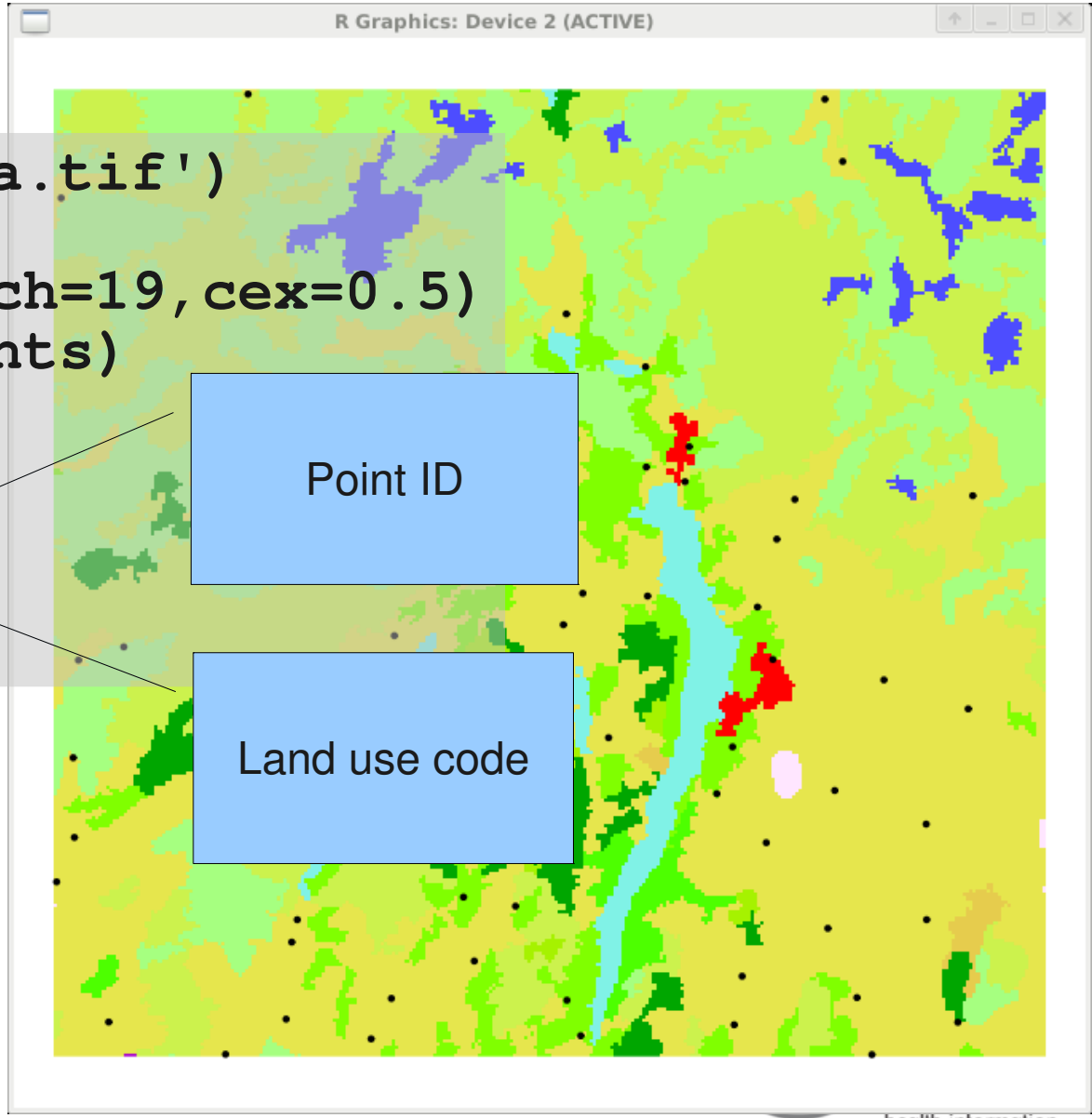
Land Use

- Continuous urban fabric
- Discontinuous urban fabric
- Industrial or commercial units
- Road and rail networks and associated land
- Port areas
- Airports
- Mineral extraction sites
- Dump sites
- Construction sites
- Green urban areas
- Sport and leisure facilities
- Non-irrigated arable land
- Permanently irrigated land
- Rice fields
- Vineyards
- Fruit trees and berry plantations
- Olive groves
- Pastures
- Annual crops associated with permanent crops
- Complex cultivation patterns
- Land principally occupied by agriculture, with sig
- Agro-forestry areas
- Broad-leaved forest
- Coniferous forest
- Mixed forest
- Natural grasslands
- Moors and heathland
- Sclerophyllous vegetatio
- Transitional woodland-sh
- Beaches, dunes, sands
- Bare rocks
- Sparsely vegetated area
- Burnt areas
- Glaciers and perpetual s
- Inland marshes
- Peat bogs
- Salt marshes
- Salines
- Intertidal flats
- Water courses
- Water bodies
- Coastal lagoons
- Estuaries
- Sea and ocean
- NODATA
- UNCLASSIFIED LAND S
- UNCLASSIFIED WATER
- UNCLASSIFIED

CHICAS
combining health information, computation and statistics

Point-raster overlay

```
> use = raster('cumbria.tif')
> plot(use)
> points(settlements,pch=19,cex=0.5)
> extract(use,settlements)
1900 1912 1931 1937
  18   18   18   18
1958 1964 1976 1980
  23   18   18   23
[etc]
```

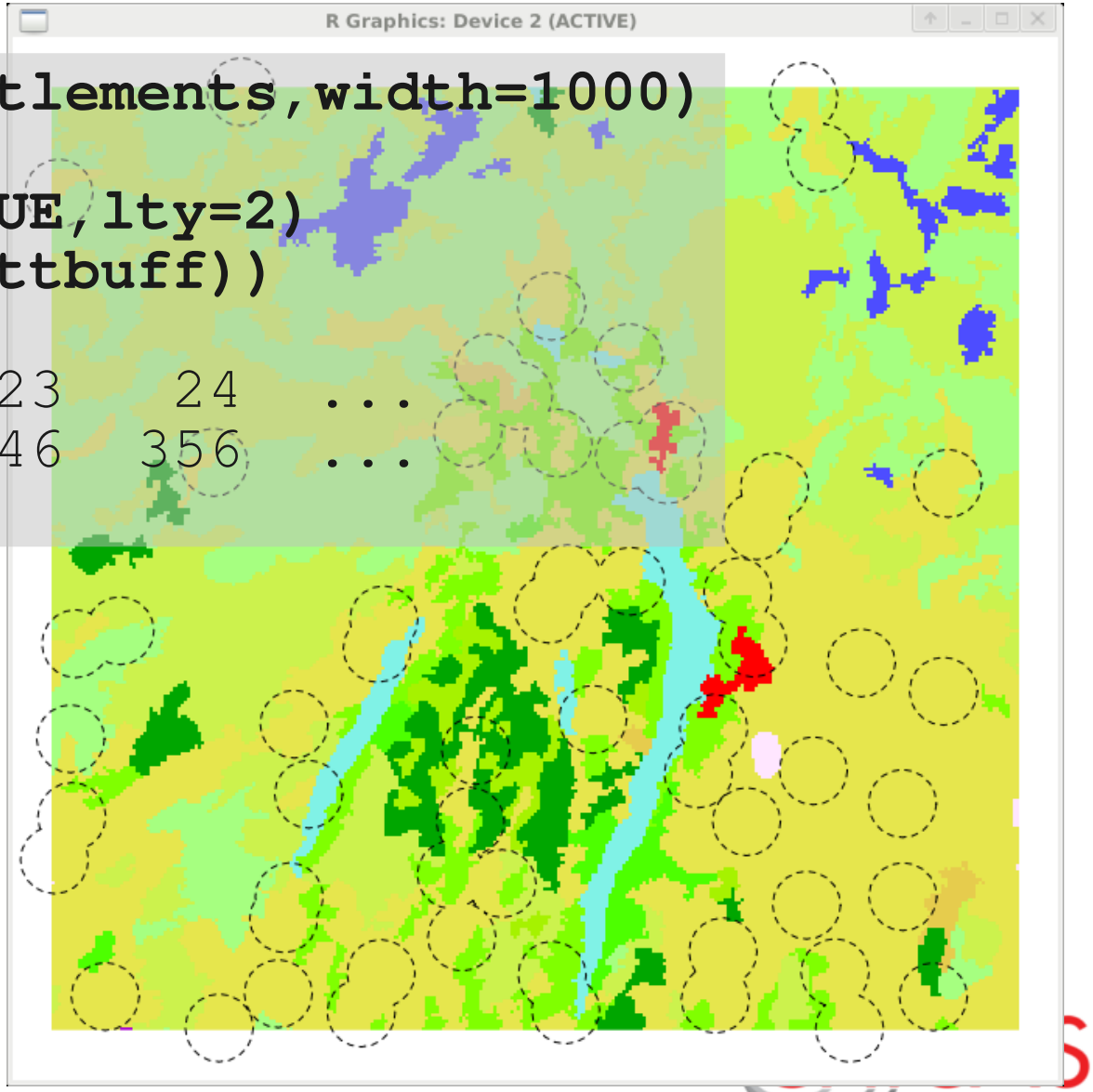


Buffering with `rgeos`

```
> settbuff=gBuffer(settlements,width=1000)
> plot(use)
> plot(settbuff,add=TRUE,lty=2)
> table(extract(use,settbuff))

   2   11   18   21   23   24  ...
 243    5 9730  173 2346  356  ...
```

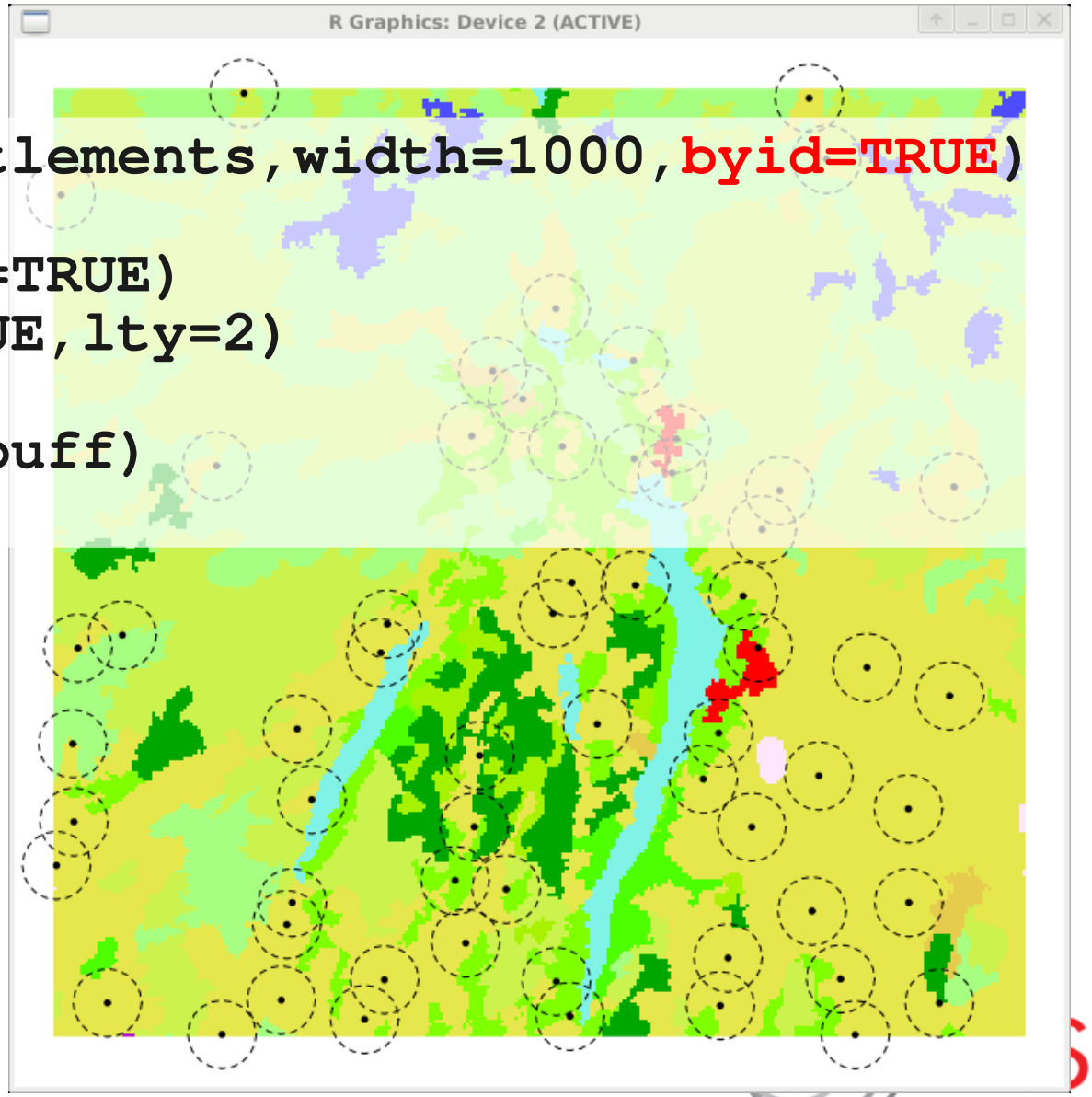
category

Number of grid cells



Per-feature buffers

```
> settbuff=gBuffer(settlements,width=1000,byid=TRUE)
> plot(use)
> plot(settlements,add=TRUE)
> plot(settbuff,add=TRUE,lty=2)

> e = extract(use,settbuff)
> et = lapply(e,table)
```



Extracting The Values

```
e = extract(use,settbuff)
```

Pixels under town 1 buffer

Pixels under town 2 buffer

```
[[1]]
  [1] 18 18 18 18 18
  [6] 18 18 18 18 18
 [11] 26 26 26 18 18
 [16] 18
[…]
[[2]]
  [1] 25 25 18 18 18
  [6] 25 25 25 18 18
 [11] 18 18 18 18 18
 [16] 18 18 25 25 25
 [21] 18 18 18 18 18
...
```

CHICAS combining health information, computation and statistics

Tabulating the values

```
et = lapply(e,table)
```

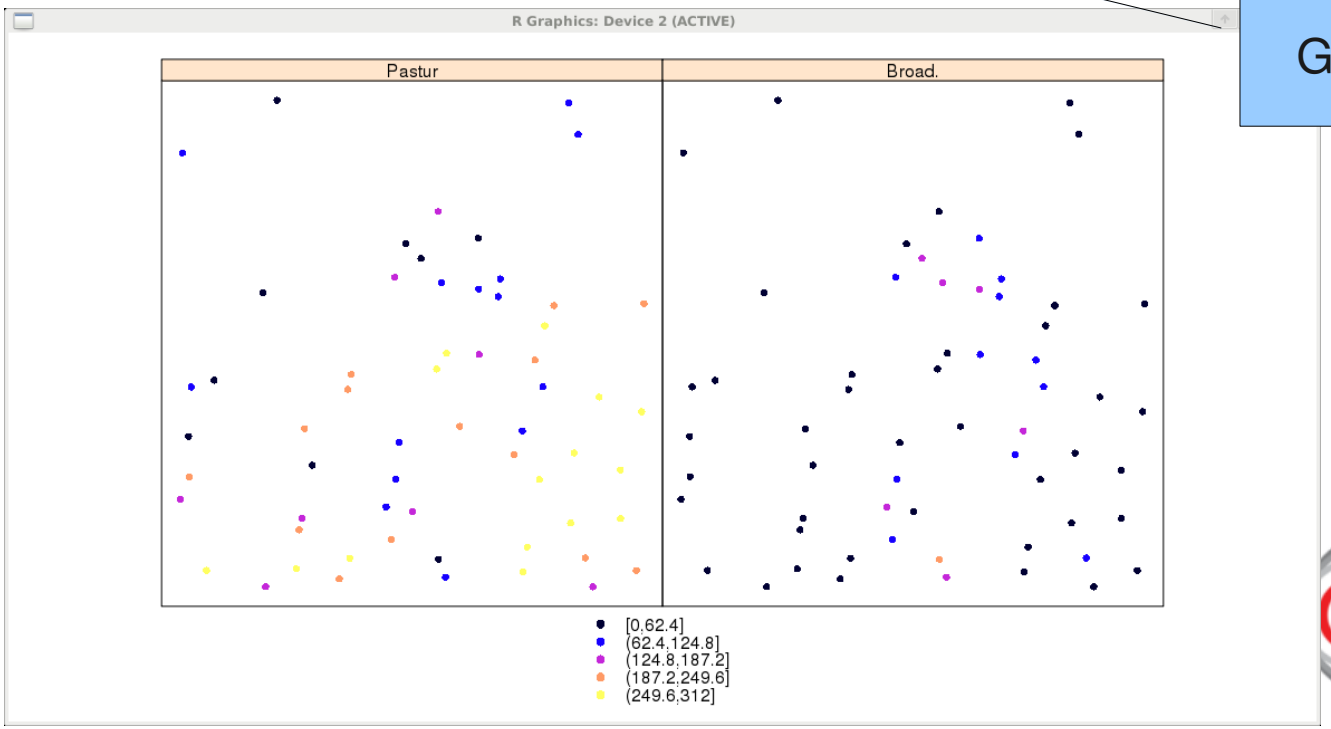
```
...
[[55]]

 18  23  26  27  41
60 112   1 109  32

[[56]]

 18  23  26  27  41
180  51   7  36  33

[[57]]

 18  26  27
 71   3 124
...
```

Category count for town 55

With a bit more massage and manipulation...

New SpatialPoints

```
> suse
          coordinates Sport. Pastur Discon Broad. Conife     NAME
1  (3471640, 3538260)      0    141       0       0       0 Gawthwaite
2  (3468280, 3539190)      0    302       0       0       0 Grizebeck
3  (3475820, 3538700)      0    224       0      28       0 Colton
4  (3473380, 3539270)      0    293       0       0       0 Lowick
5  (3481840, 3538800)      0     73       0     158       1 Lakeside
6  (3476400, 3539870)      0    311       0       0       0 Oxen Park
7  (3490200, 3538250)      0    131       0      34       0 The Howe

> spplot(suse,c('Pastur','Broad.'))
```



Grid cell counts

CHICAS
combining health information, computation and statistics

Question

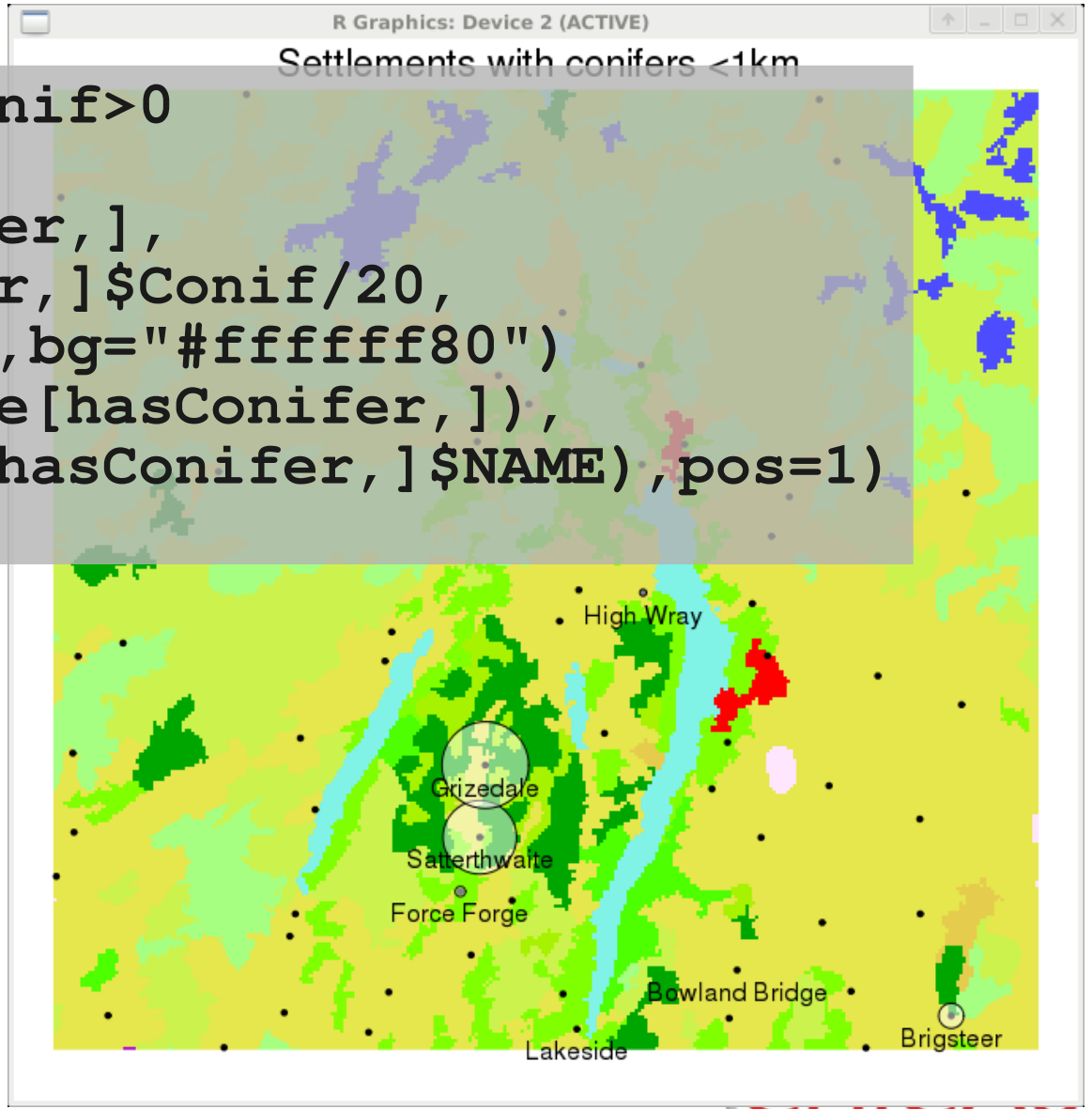
- Find all the towns that have any conifer forest within 1km
- Make a map showing how much conifer forest is around each town

LANCASTER UNIVERSITY

CHICAS
combining health information, computation and statistics

Map presentation

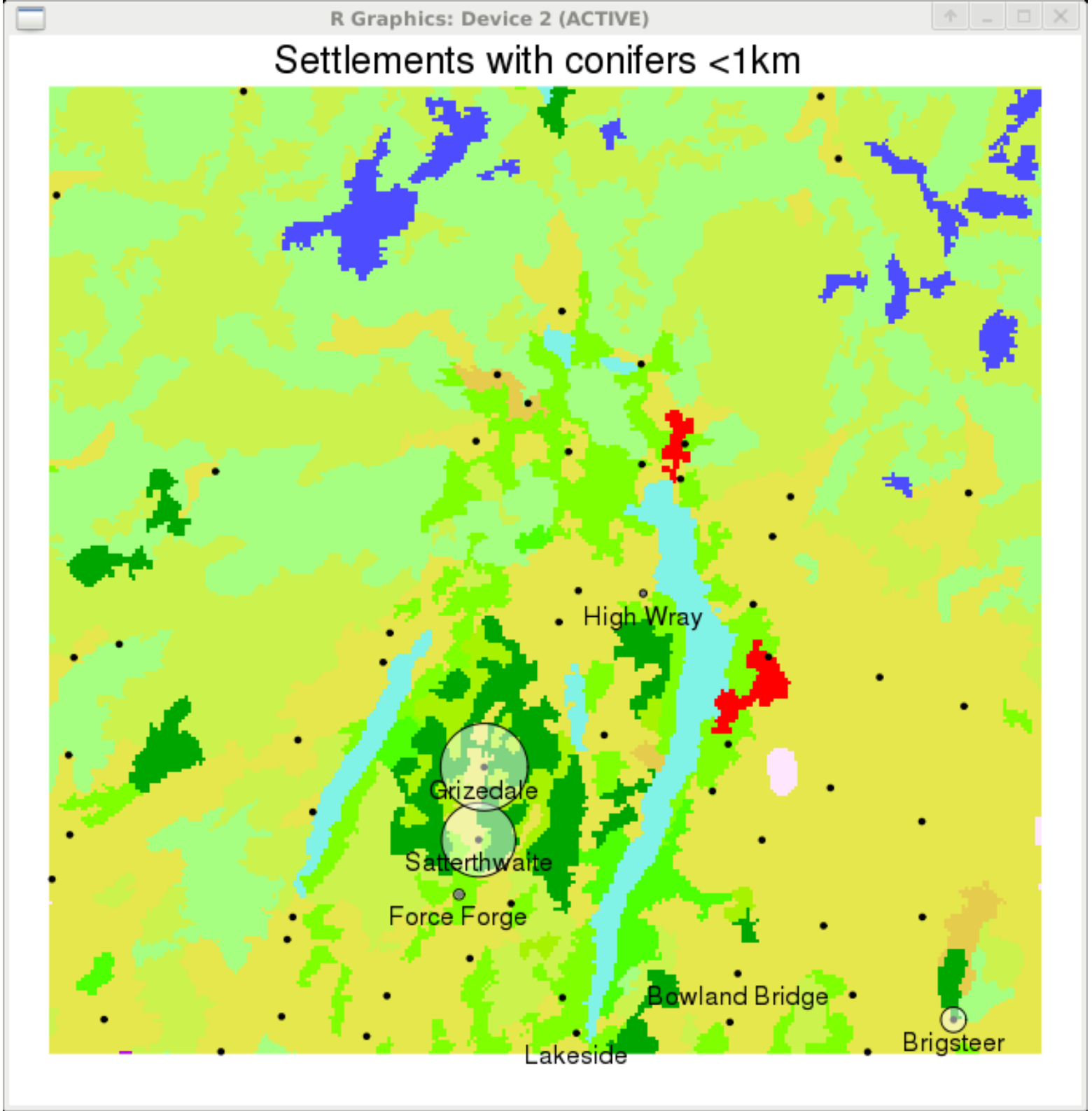
```
> hasConifer = suse$Conif>0
> plot(use)
> points(suse[hasConifer,],
    cex=suse[hasConifer,]$Conif/20,
    pch=21,col="black",bg="#ffffff80")
> text(coordinates(suse[hasConifer,]),
    as.character(suse[hasConifer,]$NAME),pos=1)
```



Automation

- Write it all up into a function

```
nearBy  = function(
        towns,
        landUse,
        code,
        dist=1000,
        thresh=0){
…
}

mapNearBy(settlements,landUse,
              22, dist=2000)
```


R Graphics: Device 2 (ACTIVE)
Settlements with conifers <1km
High Wray
Grizedale
Satterthwaite
Force Forge
Bowland Bridge
Brigsteer
Lakeside

End of Part One

- Stay tuned for more!

CHICAS
combining
health information,
computation and statistics