Lab 3: Map-making in R

Peter Diggle & Emanuele Giorgi

Lancaster Medical School, Lancaster University, Lancaster, UK

Model-based geostatistics: geospatial statistical methods for public health applications, 5-9 October 2015

Overview

- 11:00-12:30. Plotting in R; introduction to raster and shape-files; projections and conversion from long/lat to UTM coordinates.

- 12:30-13:30. Lunch.

- 13:30-15:00. Loading, plotting and extraction of information from raster files; loading and plotting shape-files; shape-files derived variables; creation of prediction grids.

R packages: `raster`, `rgdal`, `splancs`, `rgeos`, `GISTools`, `OpenStreetMap`.

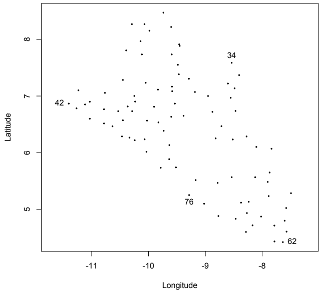
Graphical procedures in R

- **High-level:** `plot`, `curve`, `hist`, `image`, `contour`, `persp`.
- **Low-level:** `lines`, `points`, `legend`, `abline`.
- **Interactive:** `locator`, `identify`.

Graphical procedures in R

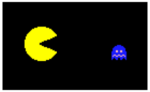
- **High-level:** `plot`, `curve`, `hist`, `image`, `contour`, `persp`.
- **Low-level:** `lines`, `points`, `legend`, `abline`.
- **Interactive:** `locator`, `identify`.

```
> data <- read.csv("http://www.lancaster.ac.uk/
                   staff/diggle/
                   Malawi2015/LiberiaRemoData.csv")
>
> plot(data[,c("long","lat")],type="n",asp=1,
+      xlab="Longitude", ylab="Latitude")
>
> points(data[,c("long","lat")],pch=20,cex=0.5)
>
> # Press esc to end the identification of points
> identify(data$long,data$lat,1:nrow(data))
[1] 34 42 62 76
```



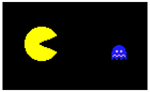
Raster files and shape files

- **Raster image:**

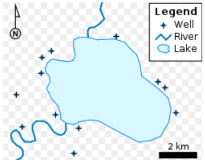


Raster files and shape files

- **Raster image:**

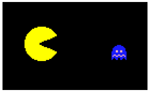


- **Shape file:**

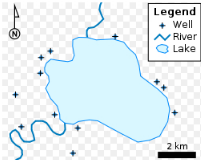


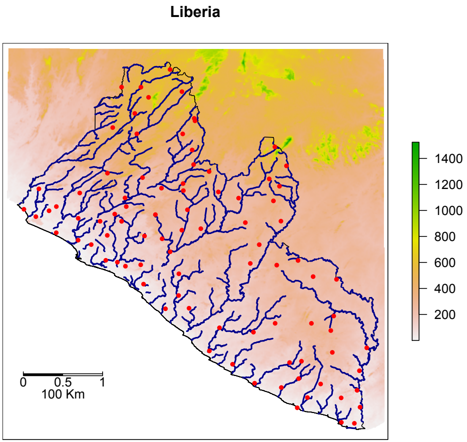
Raster files and shape files

- **Raster image:**

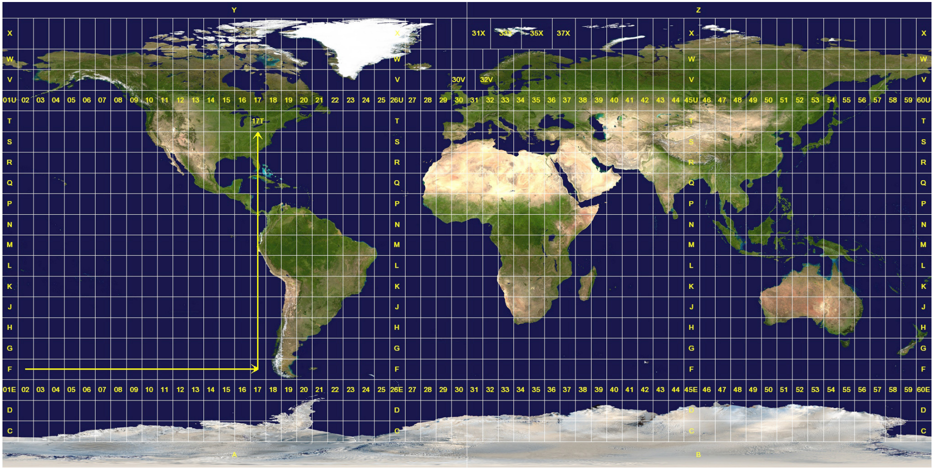


- **Shape file:**

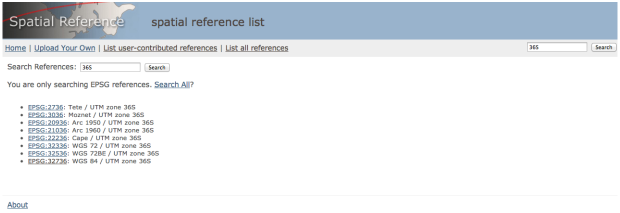




Universal Transverse Mercator (UTM) coordinate system



http://spatialreference.org/ref/epsg/


Spatial Reference
spatial reference list
Home | Upload Your Own | List user-contributed references | List all references
36S
Search

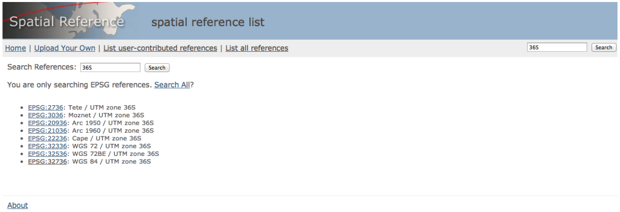
Search References: 36S Search

You are only searching EPSG references. Search All?

- EPSG:2736: Tete / UTM zone 36S
- EPSG:3036: Moznet / UTM zone 36S
- EPSG:20936: Arc 1950 / UTM zone 36S
- EPSG:21036: Arc 1960 / UTM zone 36S
- EPSG:22236: Cape / UTM zone 36S
- EPSG:32336: WGS 72 / UTM zone 36S
- EPSG:32536: WGS 72BE / UTM zone 36S
- EPSG:32736: WGS 84 / UTM zone 36S

About

http://spatialreference.org/ref/epsg/


Spatial Reference
Home | Upload Your Own | List user-contributed references | List all references
36S
Search

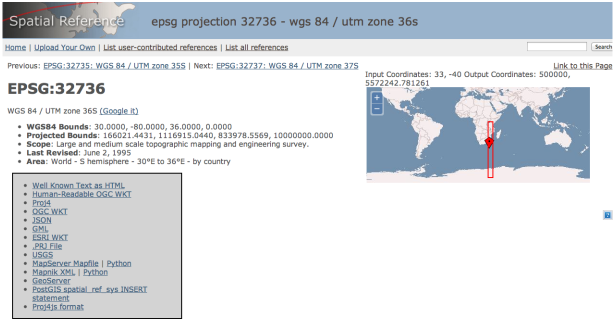
spatial reference list

Search References: 36S Search

You are only searching EPSG references. Search All?

- EPSG:2736: Tete / UTM zone 36S
- EPSG:3036: Moznet / UTM zone 36S
- EPSG:20936: Arc 1950 / UTM zone 36S
- EPSG:21036: Arc 1960 / UTM zone 36S
- EPSG:22236: Cape / UTM zone 36S
- EPSG:32336: WGS 72 / UTM zone 36S
- EPSG:32536: WGS 72BE / UTM zone 36S
- EPSG:32736: WGS 84 / UTM zone 36S

About


Spatial Reference
Home | Upload Your Own | List user-contributed references | List all references
Search
Link to this Page
+
−

epsg projection 32736 - wgs 84 / utm zone 36s

Previous: EPSG:32735: WGS 84 / UTM zone 35S | Next: EPSG:32737: WGS 84 / UTM zone 37S

Input Coordinates: 33, -40 Output Coordinates: 500000, 5572242.781261

EPSG:32736

WGS 84 / UTM zone 36S (Google it)

- **WGS84 Bounds**: 30.0000, -80.0000, 36.0000, 0.0000
- **Projected Bounds**: 166021.4431, 1116915.0440, 833978.5569, 10000000.0000
- **Scope**: Large and medium scale topographic mapping and engineering survey.
- **Last Revised**: June 2, 1995
- **Area**: World - S hemisphere - 30°E to 36°E - by country

- Well Known Text as HTML
- Human-Readable OGC WKT
- Proj4
- OGC WKT
- JSON
- GML
- ESRI WKT
- .PRJ File
- USGS
- MapServer Mapfile | Python
- Mapnik XML | Python
- GeoServer
- PostGIS spatial_ref_sys INSERT statement
- Proj4js format

UTM conversion in R

```
> library(rgdal)
> coords.longlat <- matrix(c(-13.98765, 33.78295,
+                            -13.99174, 33.79325,
+                            -13.98403, 33.78200,
+                            -13.98174, 33.77347,
+                            -13.98910, 33.78897),byrow=TRUE,
+                            ncol=2,nrow=5)
> coords.longlat <- SpatialPoints(coords.mlw,
+                    proj4string=CRS("+init=epsg:4236"))
> coords.longlat
class       : SpatialPoints
features    : 5
extent      : -13.99174, -13.98174, 33.77347, 33.79325  (xmin, xmax, ymin, ymax)
coord. ref. : +init=epsg:4236 +proj=longlat +ellps=intl
                   +towgs84=-637,-549,-203,0,0,0,0 +no_defs
>
> coords.utm <- spTransform(coords.longlat,
+                CRS("+init=epsg:32736"))
> coords.utm
class       : SpatialPoints
features    : 5
extent      : -4005723, -4005307, 14927161, 14930135  (xmin, xmax, ymin, ymax)
coord. ref. : +init=epsg:32736 +proj=utm +zone=36 +south +datum=WGS84
                   +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
```

UTM conversion in R

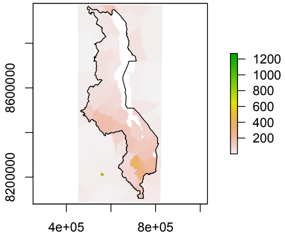
```
> library(rgdal)
> coords.longlat <- matrix(c(-13.98765, 33.78295,
+                           -13.99174, 33.79325,
+                           -13.98403, 33.78200,
+                           -13.98174, 33.77347,
+                           -13.98910, 33.78897),byrow=TRUE,
+                           ncol=2,nrow=5)
> coords.longlat <- SpatialPoints(coords.mlw,
+                   proj4string=CRS("+init=epsg:4236"))
> coords.longlat
class       : SpatialPoints
features    : 5
extent      : -13.99174, -13.98174, 33.77347, 33.79325  (xmin, xmax, ymin, ymax)
coord. ref. : +init=epsg:4236 +proj=longlat +ellps=intl
                   +towgs84=-637,-549,-203,0,0,0,0 +no_defs
>
> coords.utm <- spTransform(coords.longlat,
+               CRS("+init=epsg:32736"))
> coords.utm
class       : SpatialPoints
features    : 5
extent      : -4005723, -4005307, 14927161, 14930135  (xmin, xmax, ymin, ymax)
coord. ref. : +init=epsg:32736 +proj=utm +zone=36 +south +datum=WGS84
                   +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
```

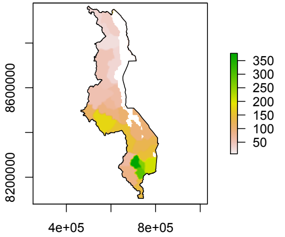
UTM conversion for Liberia

Read the data-frame `LiberiaRemoData.csv`. Convert `long` and `lat` into the appropriate UTM system using WGS84 as datum.

Raster files and shape files in R

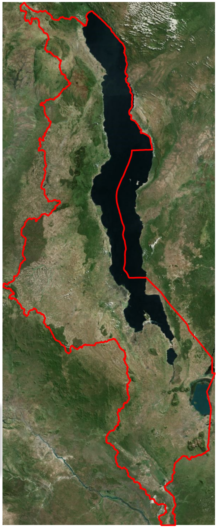
```
> par(mfrow=c(2,1),mar=c(2,2,2,2))
> Malawi.pop <- raster("Malawi/MWI_pop/mwi_pop.gri")
> Malawi.pop <- projectRaster(Malawi.pop,
+                           crs=CRS("+init=epsg:32736"))
>
> Malawi.bndrs <- readOGR("Malawi/MWI_adm/MWI_adm0.shp",
+                         "MWI_adm0")
OGR data source with driver: ESRI Shapefile
Source: "Malawi/MWI_adm/MWI_adm0.shp", layer: "MWI_adm0"
with 1 features
It has 70 fields
> Malawi.bndrs <- spTransform(Malawi.bndrs,
+                           CRS("+init=epsg:32736"))
>
> plot(Malawi.pop)
> plot(Malawi.bndrs,add=TRUE)
>
> Malawi.pop.bndrs <- mask(Malawi.pop,Malawi.bndrs)
> plot(Malawi.pop.bndrs)
> plot(Malawi.bndrs,add=TRUE)
```





Adding context

```
> library(OpenStreetMap)
> Malawi.bndrs.longlat <- spTransform(Malawi.bndrs,
+     CRS("+init=epsg:4236"))
> ul <- c(bbox(Malawi.bndrs.longlat)[2,2],
+         bbox(Malawi.bndrs.longlat)[1,1])
>
> lr <- c(bbox(Malawi.bndrs.longlat)[2,1],
+         bbox(Malawi.bndrs.longlat)[1,2])
>
> Malawi.openmap <- openmap(ul,lr,type="bing")
> plot(Malawi.openmap)
> plot(spTransform(Malawi.bndrs.longlat,osm()),add=TRUE,lwd=2,
+      border=2)
```

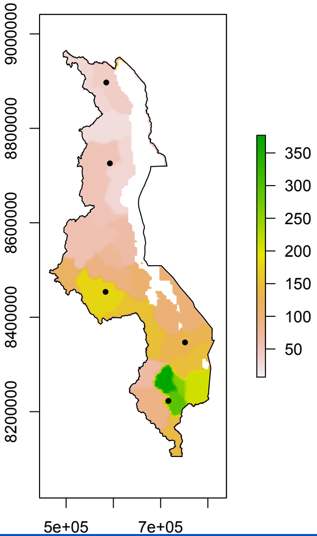


Home sweet home

Plot a map of your home-town.

Extracting information from raster files

```
> par(mfrow=c(1,1))
> loc <- matrix(c(584966.6, 8897111,
+                 592611.1, 8725874,
+                 583437.7, 8453730,
+                 751616.6, 8346707,
+                 716452.0, 8222866),ncol=2,nrow=5,byrow=TRUE)
> plot(Malawi.pop.bndrs)
> plot(Malawi.bndrs,add=TRUE)
> points(loc,pch=20)
>
> pop.loc <- extract(Malawi.pop.bndrs,loc)
> pop.loc
[1]  40.0000  50.0000 179.0000 130.0000 292.4322
>
> pop.loc.buff <- extract(Malawi.pop.bndrs,loc,buffer=50000,
+                         fun=max)
> pop.loc.buff
[1] 168  50 179 151 377
```

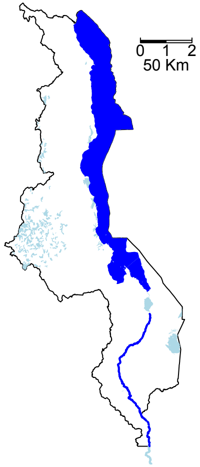


Plotting shape-files (1)

```
> Malawi.lakes <- readOGR("Malawi/MWI_wat/MWI_water_areas_dcw.shp",
+                "MWI_water_areas_dcw")
OGR data source with driver: ESRI Shapefile
Source: "Malawi/MWI_wat/MWI_water_areas_dcw.shp", layer: "MWI_water_areas_dcw"
with 182 features
It has 5 fields
> Malawi.rivers <- readOGR("Malawi/waterways/waterways.shp",
+                    "waterways")
OGR data source with driver: ESRI Shapefile
Source: "Malawi/waterways/waterways.shp", layer: "waterways"
with 847 features
It has 4 fields
> Malawi.lakes <- spTransform(Malawi.lakes,CRS("+init=epsg:32736"))
> Malawi.rivers <- spTransform(Malawi.rivers,CRS("+init=epsg:32736"))
>
> plot(Malawi.bndrs)
> plot(Malawi.lakes,col="light blue",add=TRUE,border=0)
>
```

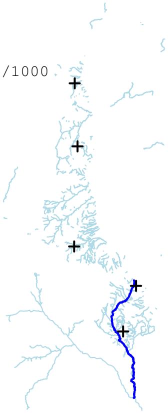
Plotting shape-files (2)

```
>
> levels(Malawi.lakes@data$NAME)
[1] "CHIA LAGOON"    "LAGO CHIUTA"    "LAKE CHIKUKUTU" "LAKE CHILWA"    "LAKE MALOMBE"
[6] "LAKE NYASA"     "MAZINZI BAY"    "SHIRE"          "UNK"
>
> lakes <- as.character(Malawi.lakes@data$NAME)
> lakes[is.na(lakes)] <- "NO-NAME"
>
> rivers <- as.character(Malawi.rivers@data$NAM)
>
> plot(Malawi.lakes[lakes=="LAKE NYASA",],
+      col="blue",border=0,add=TRUE)
> plot(Malawi.lakes[lakes=="MAZINZI BAY",],
+      col="blue",border=0,add=TRUE)
>
> Shire <- Malawi.rivers[Malawi.rivers@data$name=="Shire",]
> plot(Shire,col="light blue",lwd=2,add=TRUE)
> plot(gIntersection(Shire,Malawi.bndrs),col="blue",
+ lwd=2,add=TRUE)
>
> library(GISTools)
> map.scale(777914.5,8887990,100000,"50 Km",2)
```



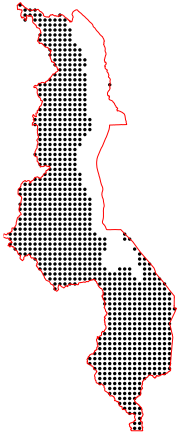
Shape-files derived variables

```
> loc <- SpatialPoints(loc,CRS("+init=epsg:32736"))
>
> plot(Malawi.rivers,col="light blue")
> plot(Shire,lwd=2,col="blue",add=TRUE)
> plot(loc,lwd=2,add=TRUE)
>
> dist.Shire.loc.km <- rep(NA,length(loc))
> dist.ww.loc.km <- rep(NA,length(loc))
>
> for(i in 1:length(loc)) dist.Shire.loc.km[i] <- gDistance(loc[i,],Shire)/1000
> for(i in 1:length(loc)) dist.ww.loc.km[i] <- gDistance(loc[i,],
+                                                 Malawi.rivers)/1000
> dist.Shire.loc.km
[1] 558.780278 394.820082 186.455302   5.931018  14.555897
>
> dist.ww.loc.km
[1] 2.48537077 6.87656881 0.03779594 5.93101841 8.48146424
```



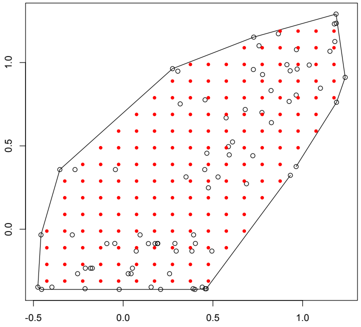
Prediction grids within a shape-file polygon

```
> library(splancs)
>
> check.inout.shapefile <- function(s.points,
+                                   shp) {
+    sapply(1:length(s.points), function(i)
+    !all(is.na(over(shp,s.points[i,],returnList=FALSE))))
+ }
>
> ext.Malawi <- as.matrix(extent(Malawi.bndrs))
> box.Malawi <- rbind(c(ext.Malawi[1,1],ext.Malawi[2,1]),
+                     c(ext.Malawi[1,2],ext.Malawi[2,1]),
+                     c(ext.Malawi[1,2],ext.Malawi[2,2]),
+                     c(ext.Malawi[1,1],ext.Malawi[2,2]),
+                     c(ext.Malawi[1,1],ext.Malawi[2,1]))
>
> grid.tot <- SpatialPoints(gridpts(box.Malawi,xs=10000,ys=10000),
+             CRS("+init=epsg:32736"))
> grid.tot.in <- check.inout.shapefile(grid.tot,
+                            Malawi.lakes[lakes=="LAKE NYASA" |
+                            lakes=="MAZINZI BAY",])
> grid.no.Lake <- grid.tot[!grid.tot.in,]
> grid.no.Lake <- SpatialPoints(grid.no.Lake,CRS("+init=epsg:32736"))
> grid.pred <- gIntersection(grid.no.Lake,Malawi.bndrs)
> plot(grid.pred,pch=20,cex=0.5)
> plot(Malawi.bndrs,add=TRUE,border=2)
```



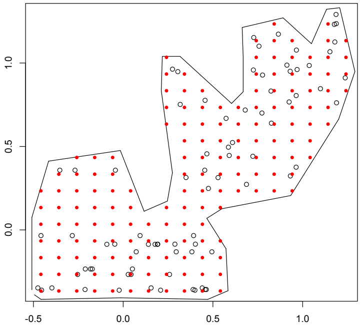
Prediction grids within polygons (1)

```
> set.seed(123)
> coords1 <- cbind(runif(40,-0.5,0.5),runif(10,-0.5,0.5))
> coords2 <- cbind(runif(20,0.7,1.3),runif(20,0.7,1.3))
> coords3 <- cbind(runif(20,0.2,1),runif(20,0.2,1))
> coords <- rbind(coords1,coords2,coords3)
>
> plot(coords,xlab="",ylab="")
> coords.chull <- coords[chull(coords),]
> lines(coords.chull,type="l")
> grid.chull <- gridpts(coords.chull,xs=0.1,ys=0.1)
> points(grid.chull,pch=20,col=2)
```



Prediction grids within polygons (2)

```
> plot(coords,xlab="",ylab="")
> poly <- locator(type="l")
> poly <- cbind(poly$x,poly$y)
> grid.poly <- gridpts(poly,xs=0.1,ys=0.1)
> points(grid.poly,pch=20,col=2)
```



Exercise

Tiyeni tipite ku Liberia!

Use the UTM coordinates obtained in Slide 6 for Liberia to carry out the following tasks.

1. Create an OpenStreet map of Liberia, using a square of 6 degrees with center in $6.3167°$ N, $9.700°$ W. Plot the waterways of Liberia, highlighting Cavally river. Plot the locations in `LiberiaRemoData.csv` on the map. Finally, compute the distance of each location from its closest waterway. Note that all the shape-files and locations should be first transformed using the `osm()` projection.
2. Plot the elevation raster file within the boundaries of Liberia, adding a map scale. Compute the mean elevation within 5 km for each of the locations in `LiberiaRemoData.csv`.
3. Create a 5 by 5 km regular grid within Liberia.