

Geospatial Data in R and Beyond



Spatial Cheatsheet

This cheatsheet is an attempt to supply you with the key functions and manipulations of spatial vector and raster data. It does not have examples for you to cut and paste, its intention is to provoke the "Oh yes, that's how you do it" thought when stuck.



Packages

```
library(sp) # vector data
library(raster) # raster data
library(rgdal) # input/output, projection
library(rgeos) # geometry ops
library(spdep) # spatial dependence
```

Links

- [Spatial Task View](#)
- [R-SIG-Geo mailing list](#)
- [Spatial at R-forge](#)
- [Spatial at R-wiki](#)
- [Spatial at Crantastic!](#)
- ["map" at RGM](#)
- [OSgeo](#)
- [StackOverflow R Questions](#)

Points

```
# points from scratch
coords = cbind(x, y)
sp = SpatialPoints(coords)
# make spatial data frame
spdf = SpatialPointsDataFrame(coords, data)
spdf = SpatialPointsDataFrame(sp, data)
# promote data frame to spatial
coordinates(data) = cbind(x, y)
coordinates(data) = ~lon + lat
# back to data
as.data.frame(data)

##      lon  lat Z
## 1 11.515 24.52 d
## 2   7.056 27.11 a
## 3 12.945 30.09 c
## 4 12.793 24.72 e
```

Lines

```
c1 = cbind(x1, y1)
c2 = cbind(x2, y2)
c3 = cbind(x3, y3)

# simple line strings
L1 = Line(c1)
L2 = Line(c2)
L3 = Line(c3)

# single/multiple line strings
Ls1 = Lines(list(L1), ID = "a")
Ls2 = Lines(list(L2, L3), ID = "b")

# with spatial nature
SL1 = SpatialLines(list(Ls1))
SL12 = SpatialLines(list(Ls1, Ls2))
```

Polygons

```
# single ring feature
c1 = cbind(x1, y1)
r1 = rbind(c1, c1[1, ]) # join
P1 = Polygon(r1)
Ps1 = Polygons(list(P1), ID = "a")

# double ring feature
c2a = cbind(x2a, y2a)
r2a = rbind(c2a, c2a[1, ])
c2b = cbind(x2b, y2b)
r2b = rbind(c2b, c2b[1, ])

P2a = Polygon(r2a)
P2b = Polygon(r2b)
Ps2 = Polygons(list(P2a, P2b), ID = "b")

# Spatial Polygons Data Frame
```

```
## 5 12.888 28.24 b

data@data

##      Z
## 1 d
## 2 a
## 3 c
## 4 e
## 5 b

bbox(spdf)

##      min      max
## x  7.056 12.94
## y 24.520 30.09
```

```
# made into spatial data frame
SLDF = SpatialLinesDataFrame(SL12, data.frame(Z = c(
  "Road", "River"), row.names = c("a",
  "b")))

as.data.frame(SLDF)

##      Z
## a  Road
## b River

SpatialLineLengths(SLDF)

## [1] 2.414 4.828
```

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

SPDF = SpatialPolygonsDataFrame(SP, data.frame(N = c("one", "two"), row.names = c("a", "b")))
SPDF@data

##      N
## a one
## b two

# single ring with hole
hc1 = cbind(xh1, yh1)
hr1 = rbind(hc1, hc1[1, ])
H1 = Polygon(hr1, hole = TRUE)
Plh = Polygons(list(P1, H1), ID = "c")
SP1h = SpatialPolygons(list(Plh))
# plot(SP1h, usePolypath=TRUE)
```

Raster

```
# from x,y,z-matrix
r1 = raster(list(x = x, y = y, z = z))
# rows and columns values
r1[, 3]

## [1] 0.7377 0.3342 0.6924 0.3482 0.2972 0.8148 0.8212 0.5362 0.8750 0.9808 0.2729

r1[2, ]

## [1] 0.40396 0.79350 0.33422 0.25095 0.64577 0.88173 0.50432 0.73244 0.98500 0.13277 0.59993 0.04035

extent(r1)

## class      : Extent
## xmin       : 22.95
## xmax       : 29.45
## ymin       : 44
## ymax       : 46.3

dim(r1)
```

Coordinates

```
# EPSG strings
latlong = "+init=epsg:4326"
ukgrid = "+init=epsg:27700"
google = "+init=epsg:3857"

# Spatial*
proj4string(SPDF)

## [1] NA

proj4string(SPDF) = CRS(latlong)
SL1 = SpatialLines(list(Ls1), proj4string = CRS(latlong))

# Raster CRS
projection(r1)

## [1] "NA"

# - assign or set on creation
projection(r1) = CRS(latlong)
r1 = raster(list(x = x, y = y,
```

I/O

```
# -- vectors

# avoid - doesn't read CRS
library(maptools)
shapes = readShapeSpatial("data.shp")

# read/write shapefiles (and others)
# - list formats
ogrDrivers()
shapes = readOGR(".", "data")
writeOGR(shapes, ".", "data", "ESRI Shapefile")
writeOGR(shapes, "towns.kml", "towns", "KML")

# -- rasters

# creates SpatialGrid objects
r = readGDAL("data.tif")

# create Rasters/Brick objects from files
```

```
## [1] 11 12 1

# create empty, then fill
r2 = raster(nrows = nrows,
ncols = ncols, xmn = xmn,
xmx = xmx, ymn = ymn, ymx
= ymx)
r2[] = runif(nrows * ncols
)

# create from extent, then
set values
r3 = raster(extent(r2), nr
ows = nrows, ncols = ncols
)
values(r3) = runif(nrows *
ncols)

# multi-band stack
s1 = stack(list(r1, r2, r3
))
dim(s1)

## [1] 11 12 3

# multi-band brick
b1 = brick(list(r1, r2, r3
))
```

```
= y, z = z), crs = latlong
)

# Transform Spatial*
SPtrans = spTransform(SPDF
, CRS(google))

# Transform/Warp Raster
rTrans = projectRaster(r1,
crs = google)
```

```
r = raster("data.tif")
# - write Raster to GeoTIF
F
writeRaster(r, "data2.tif"
, "GTiff")
# - supported formats
writeFormats()
# or for Google Earth
KML(r, "r.kmz")
```

Manipulation

```
# Spatial*DataFrames
#
# subset(Towns,pop>29000) doesn't work
BigTowns = subset(Towns, Towns$pop > 29000
)
BigTowns = Towns[Towns$pop > 29000, ]
BigAndSmall = rbind(BigTowns, SmallTowns)

# points in zones
Towns[1:10, ] %over% Zones

## [1] 1 NA 2 NA NA 2 NA NA NA 1

# rasters
# - sample points
vpt = extract(r1, sPoints)
# - sample polygons
vpoly = extract(r1, sPolys)
# - crop an area
e = extent(raster(xmn = 25, xmx = 27, ymn
= 44.5, ymx = 45.5))
cr1 = crop(r1, e)
```

Plotting

```
# scale colour
library(RColorBrewer)
palette(brewer.pal(6, "YlOrRd"))
plot(Towns, col = plotrix::rescale(Towns$
pop, c(1, 6)), pch = 19)

# scale size
plot(Towns, cex = plotrix::rescale(Towns$
pop, c(1, 4)), pch = 19)

# polygons
plot(Zones, col = fillColour, border = out
lineColour)

# sp colours
sp.theme(set = TRUE, regions = list(col =
colours))
spplot(Towns, "pop")

# rasters
plot(r1)
```

```
# true colour from bricks  
plotRGB(b1, scale = 1)
```

© Barry Rowlingson 2012 | stock imagery from [Stock.xchng](#)



UseR 2012 Spatial Data Workshop is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#).