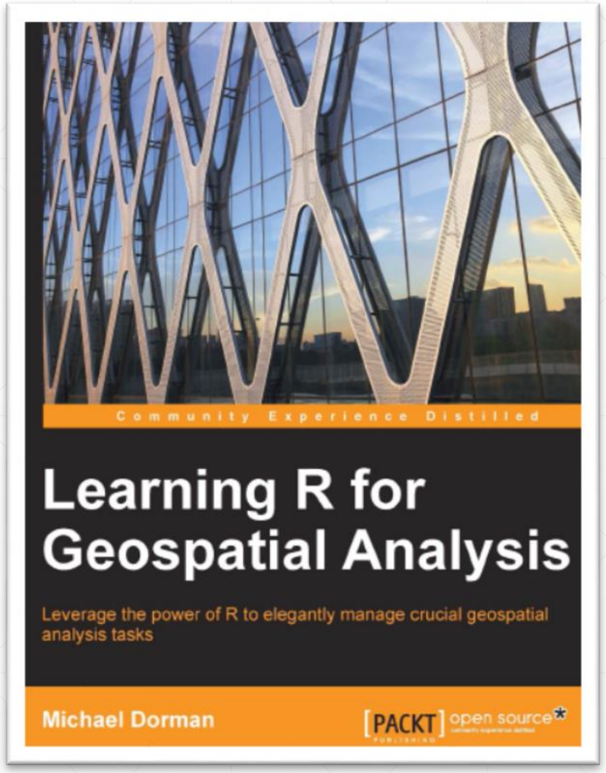
Introduction to spatial data in R

Introduction to spatial data in R

- Introduction to spatial data in R
- Packages for spatial data
- Classes for vector and raster datasets
- Reading and writing of spatial data

Books

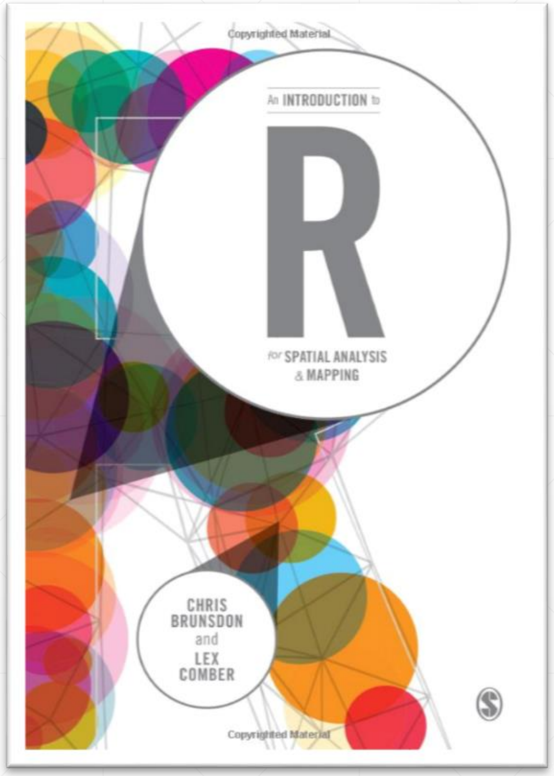

Community Experience Distilled

Learning R for Geospatial Analysis

Leverage the power of R to elegantly manage crucial geospatial analysis tasks

Michael Dorman

[PACKT] open source*


Copyrighted Material

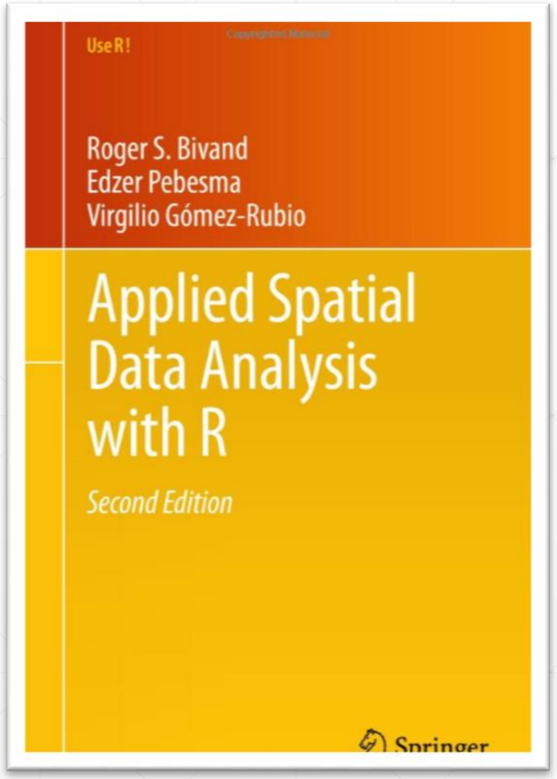
An INTRODUCTION to

R

for SPATIAL ANALYSIS & MAPPING

CHRIS BRUNSDON and LEX COMBER

Copyrighted Material


Use R!

Roger S. Bivand
Edzer Pebesma
Virgilio Gómez-Rubio

Applied Spatial Data Analysis with R

Second Edition

Springer

Links

Overview of R packages for spatial data:
http://cran.r-project.org/web/views/Spatial.html

Description of the sp package:
http://cran.r-project.org/web/packages/sp/sp.pdf

Description of the raster package:
http://cran.r-project.org/web/packages/raster/vignettes/Raster.pdf

Spatial-analyst.net – a Wiki for spatial data analysis based on R:
http://spatial-analyst.net/

Some more tips:
http://spatialanalysis.co.uk/r/

Mailinglist of R special interest group on using geographical data and mapping
https://stat.ethz.ch/mailman/listinfo/R-SIG-Geo/

Packages for spatial data in R

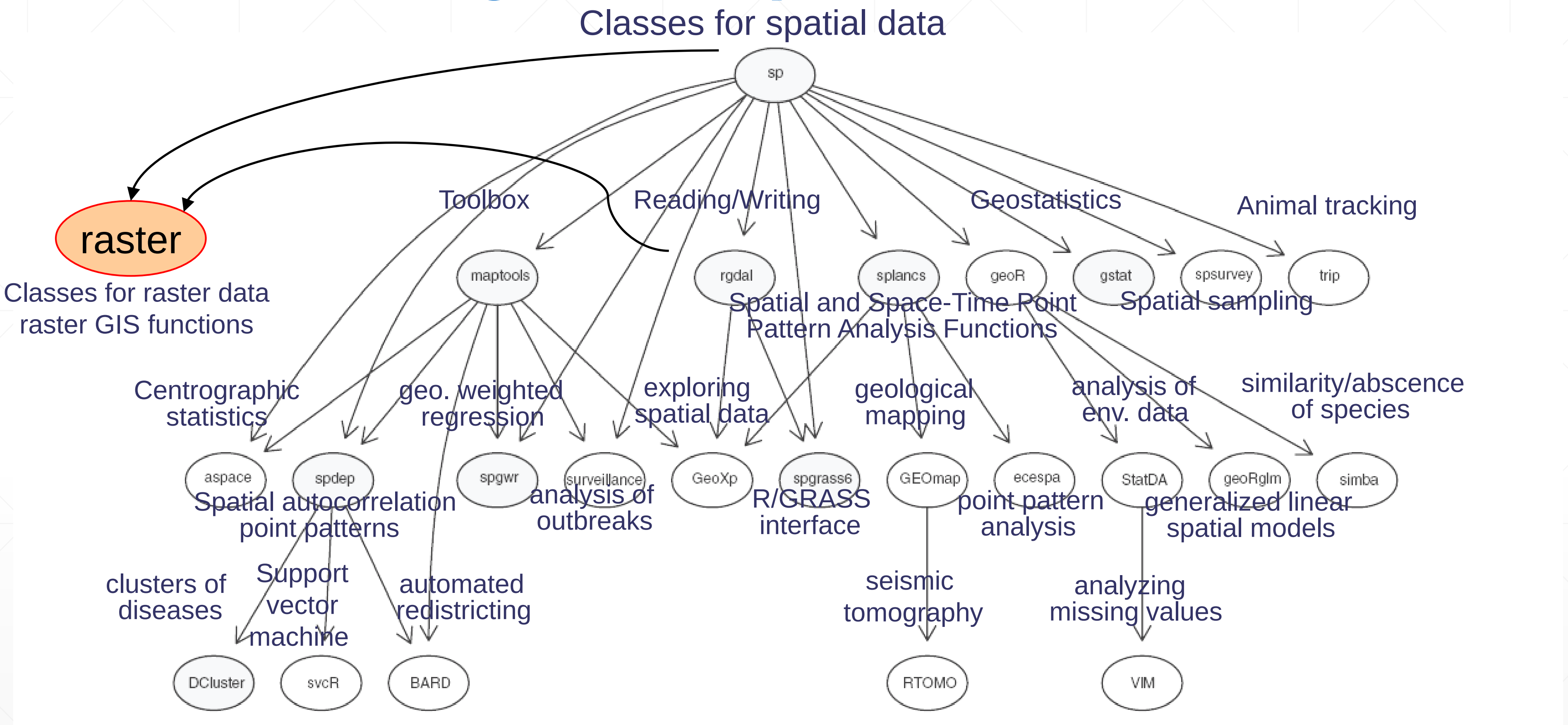
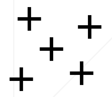


Fig. 1.1. Tree of R contributed packages on CRAN depending on or importing sp directly or indirectly; others suggest sp or use it without declaration in their package descriptions (status as of 2008-04-06)

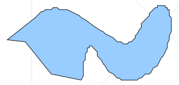
Fig.: R packages depending on package sp (Bivand et al. 2008:5)
+ raster package for geographic analysis and modeling with raster data

Types of spatial data

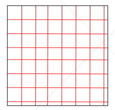
Vector data

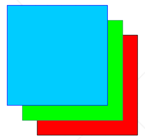
Point 

Line

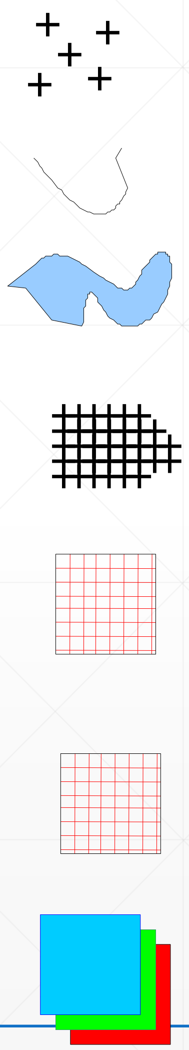
Polygon 

Raster data

 Rasterlayer

 Rasterstack

Classes for spatial data in R



Data type	read/write	classes from package sp	Basic methods for classes

Vector data	readOGR()
	writeOGR()
		SpatialPoints
		SpatialPointsDataFrame
		SpatialLines
		SpatialLinesDataFrame
		SpatialPolygons
		SpatialPolygonsDataFrame
			Get extent:
			bbox()
			Get projection:
			proj4string()
			Get coordinates:
			coordinates()
			Access data:
			@data

Raster data		SpatialPixels SpatialPixelsDataFrame	

Raster data	readGDAL()
	writeGDAL()
		SpatialGrid
		SpatialGridDataFrame	

classes from package raster:

Data type	read/write	classes	Basic methods
Raster data	raster()	RasterLayer	Get extent: extent() Get resolution: res() Get projection: projection() Get data: getValues()

Raster data	stack()
	brick()
		RasterStack – multiple files
		RasterBrick – one file	

Classes for vector data: Spatial* classes

SpatialPoints, SpatialLines, SpatialPolygons, SpatialPixels

Read ESRI shapefile with country borders:

```
> admin <- readOGR(".", "110m_admin_0_countries")
OGR data source with driver: ESRI Shapefile
Source: ".", layer: "110m_admin_0_countries"
with 177 features and 24 fields
Feature type: wkbPolygon with 2 dimensions
```

```
> bbox(admin)
   min       max
x -180 180.00000
y  -90  83.64513
> proj4string(admin)
[1] " +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +to
```

```
> x <- admin@data
> is.data.frame(x)
[1] TRUE
```

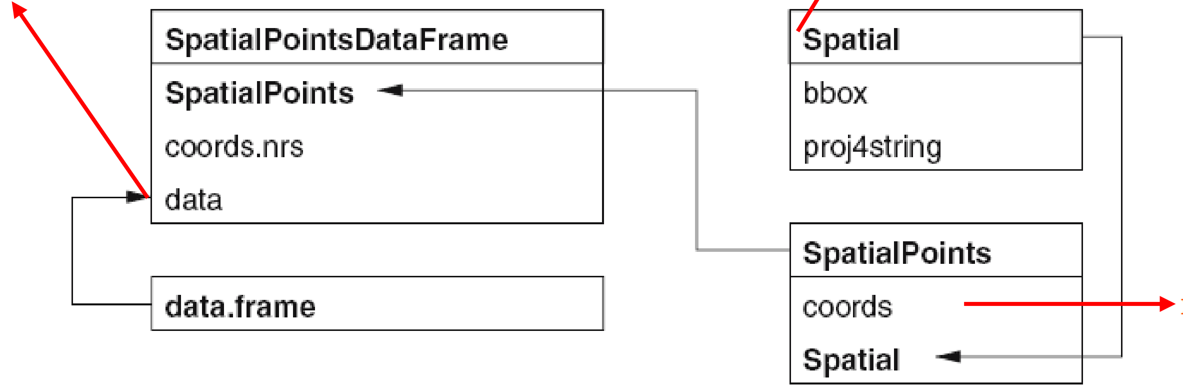
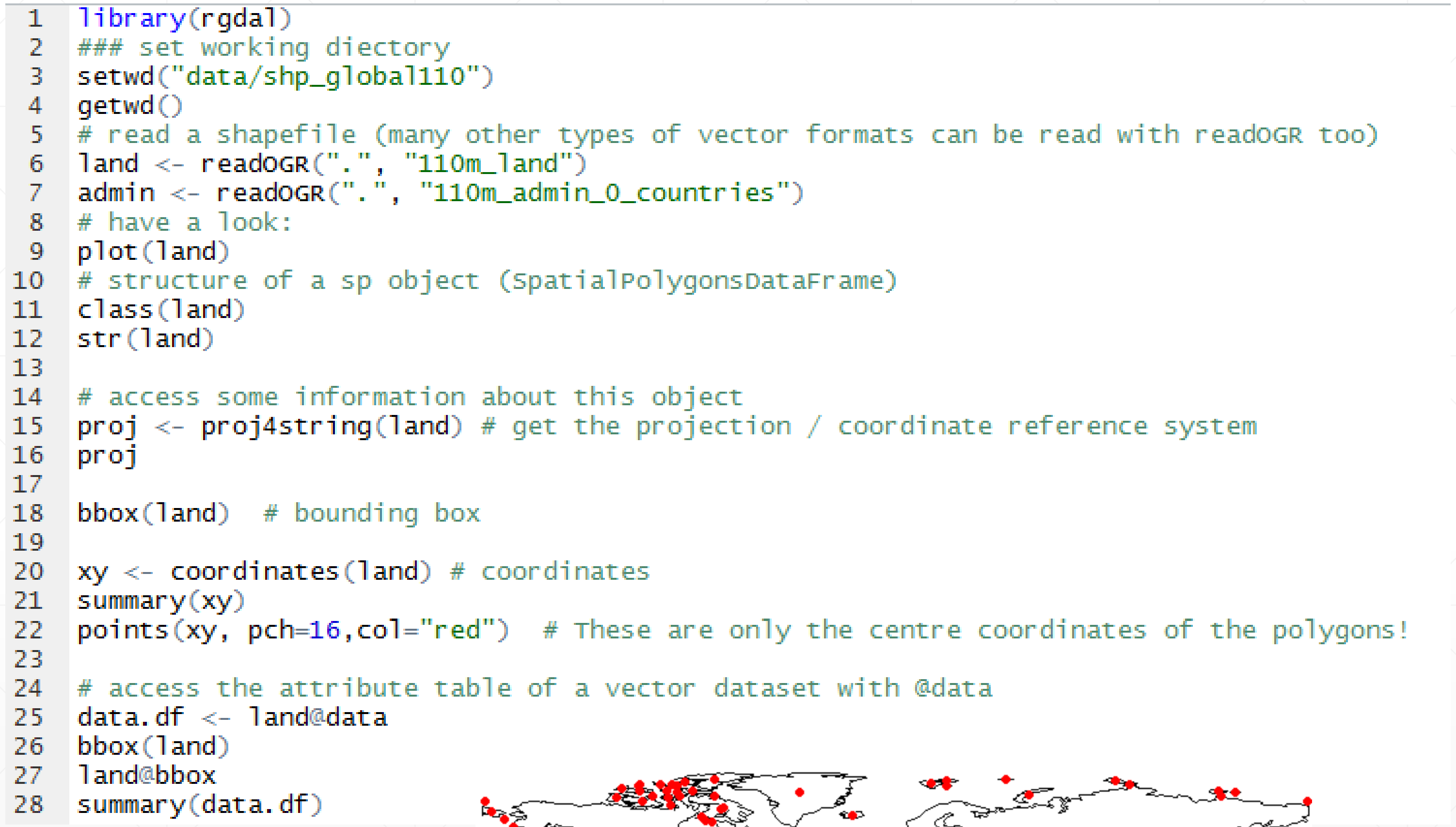


Fig. 2.2. Spatial points classes and their slots; arrows show subclass extensi

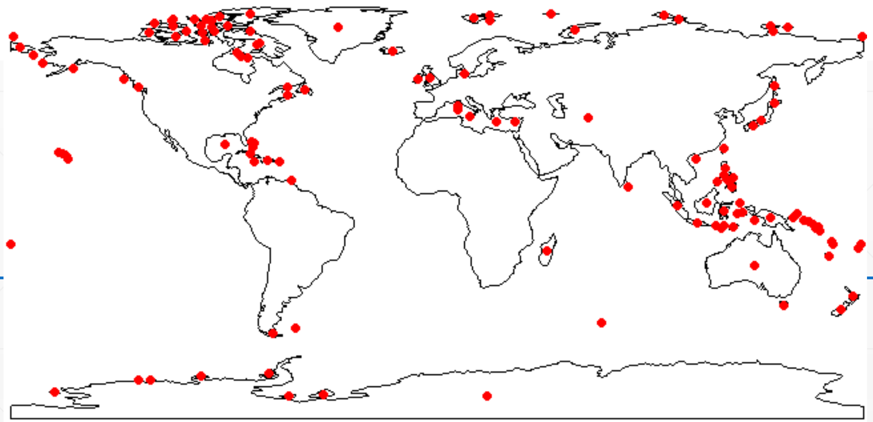
```
> coordinates(admin)
            [,1]        [,2]
[1,]   66.086690  33.8563993
[2,]   17.502912 -12.2915534
[3,]   20.032426  41.1413533
[4,]   54.206715  23.8686337
[5,]  -65.149543 -35.2201720
[6,]   45.000290  40.2166076
[7,]   21.284393 -80.5227822
[8,]   69.531580 -49.3064549
```

Fig.: Class hierarchy for Spatial Points (Bivand et al. 2008:35)

Read shapefile

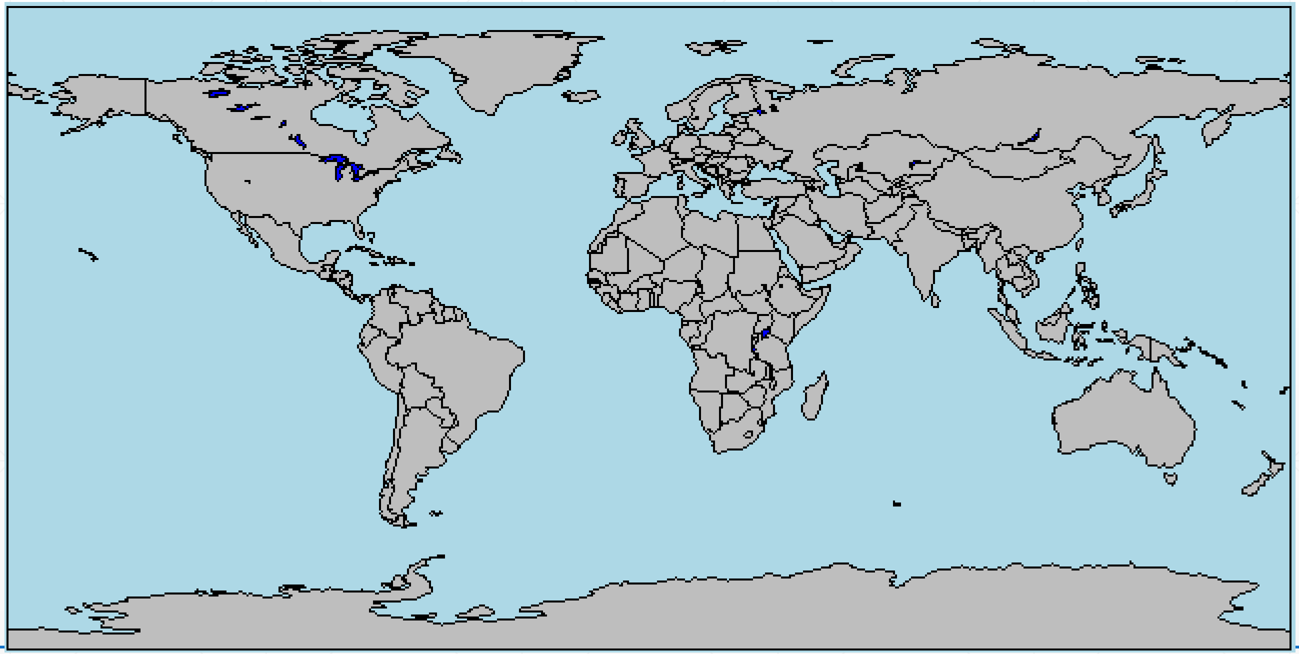


```
library(rgdal)
### set working diectory
setwd("data/shp_global110")
getwd()
# read a shapefile (many other types of vector formats can be read with readOGR too)
land <- readOGR(".", "110m_land")
admin <- readOGR(".", "110m_admin_0_countries")
# have a look:
plot(land)
# structure of a sp object (SpatialPolygonsDataFrame)
class(land)
str(land)

# access some information about this object
proj <- proj4string(land) # get the projection / coordinate reference system
proj

bbox(land)  # bounding box

xy <- coordinates(land) # coordinates
summary(xy)
points(xy, pch=16,col="red")  # These are only the centre coordinates of the polygons!

# access the attribute table of a vector dataset with @data
data.df <- land@data
bbox(land)
land@bbox
summary(data.df)
```

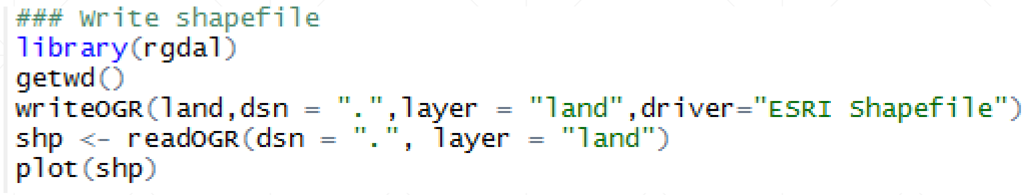


Plot shapefile

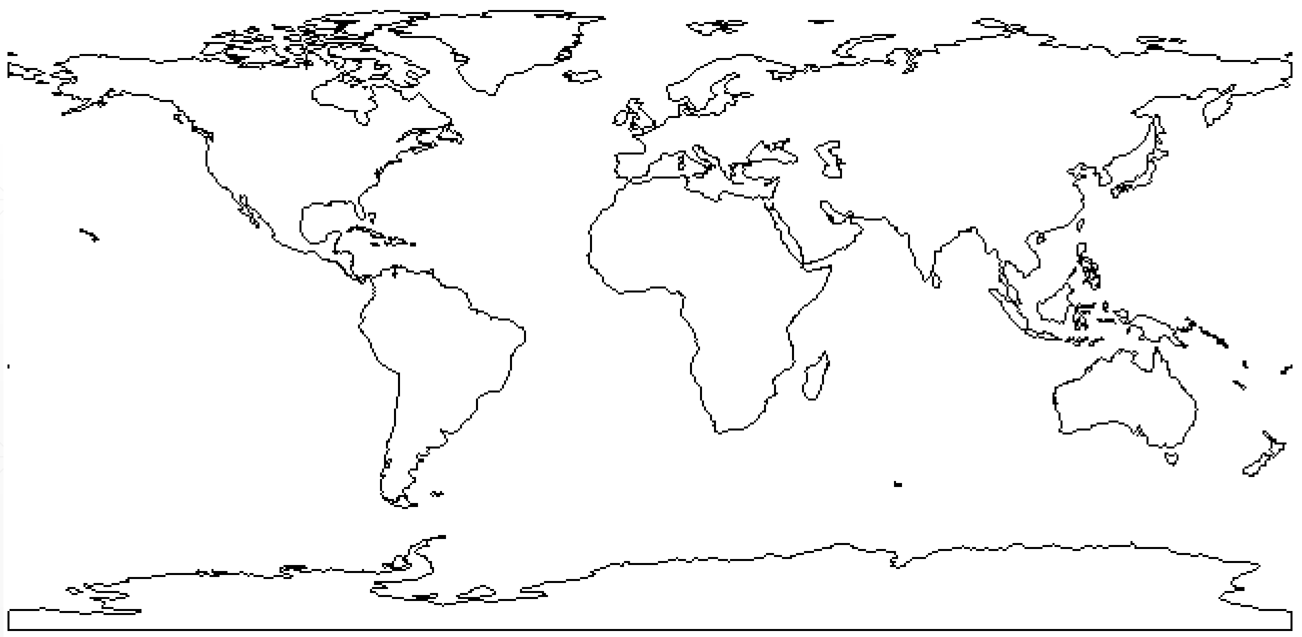
```
files <- list.files(pattern=".shp")
files # filenames of the files to be read

world <- readOGR(".","110m_land")
plot(world,col="grey",border="blue",bg = "lightblue")
plot(admin,add=TRUE)

lakes <- readOGR(".","110m_lakes")
plot(lakes,add=TRUE,col = "blue")

ocean <- readOGR(".","110m_ocean")
plot(ocean,add=TRUE,col = "lightblue")
```



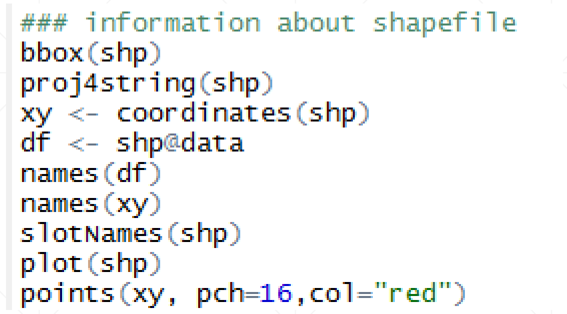
Write shapefile



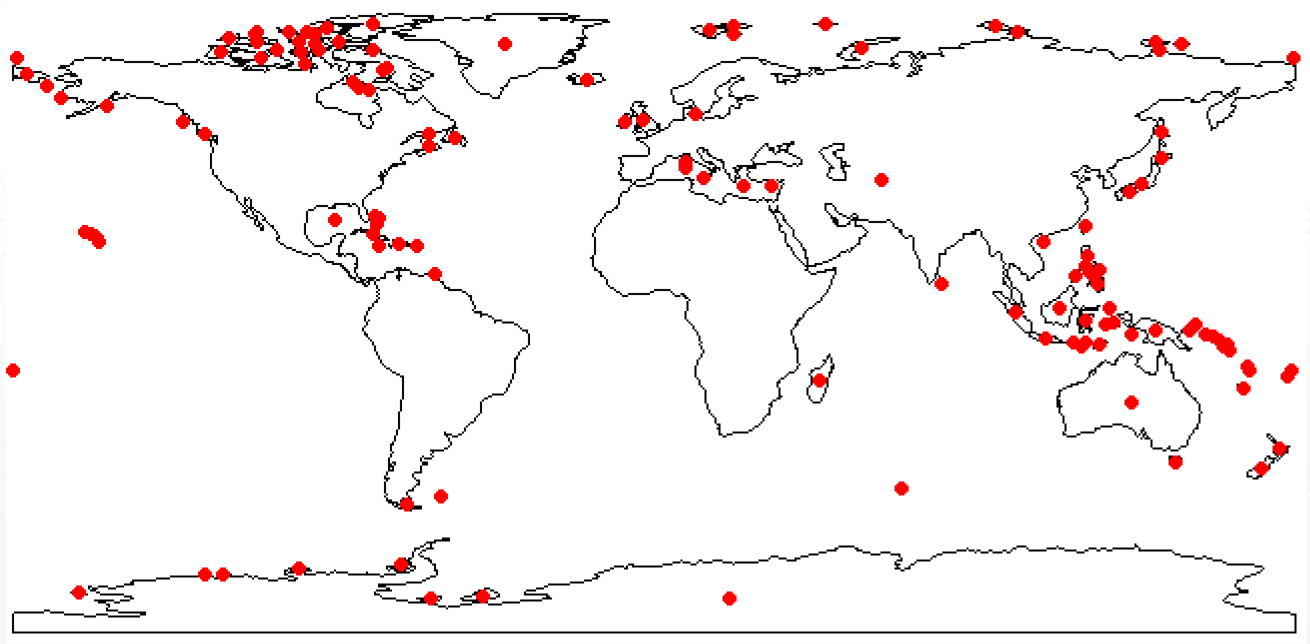
```
### Write shapefile
library(rgdal)
getwd()
writeOGR(land,dsn = ".",layer = "land",driver="ESRI Shapefile")
shp <- readOGR(dsn = ".", layer = "land")
plot(shp)
```



Shapefile properties

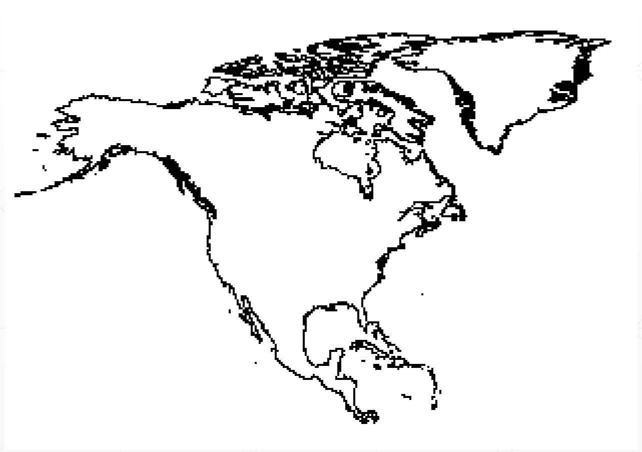


```
### information about shapefile
bbox(shp)
proj4string(shp)
xy <- coordinates(shp)
df <- shp@data
names(df)
names(xy)
slotNames(shp)
plot(shp)
points(xy, pch=16,col="red")
```



Select by attribute

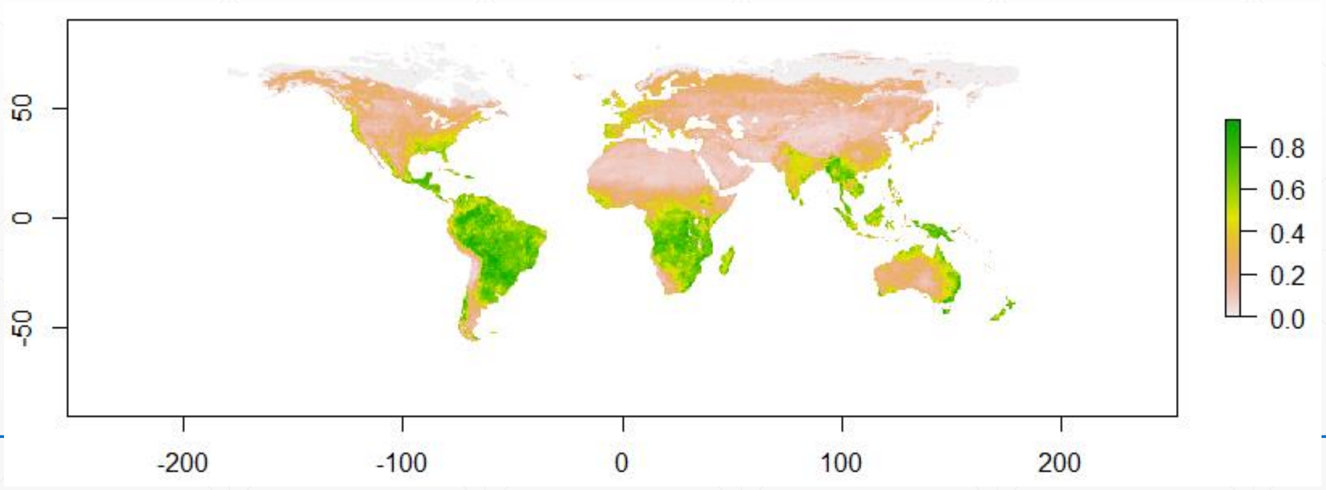
```
library(rgdal)
folder <- "C:\\Program Files (x86)\\ArcGIS\\Desktop10.3\\ArcGlobeData"
shp <- readOGR(dsn = folder,layer = "continent")
plot(shp,col="grey")

plot(shp,lwd=2,add=TRUE)
df <- shp@data

sel <- df$CONTINENT == "North America"
shp[sel,]
plot(shp[sel,])
```



Raster properties

- **Single-band RasterLayer**

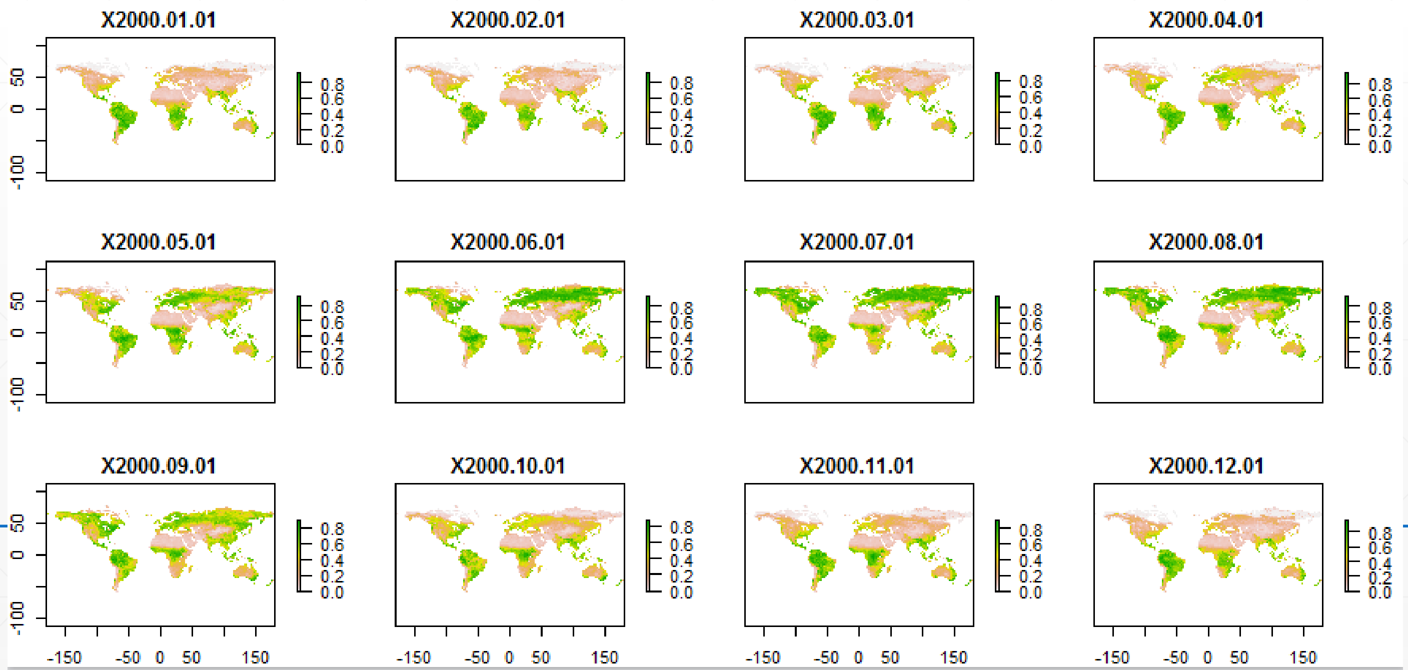
```
nrow(ndvi)
ncol(ndvi)
ncell(ndvi)
extent(ndvi)
bbox(ndvi)
res(ndvi)
projection(ndvi)
```



Read Multi-band Raster

➢ Multi-band RasterBrick

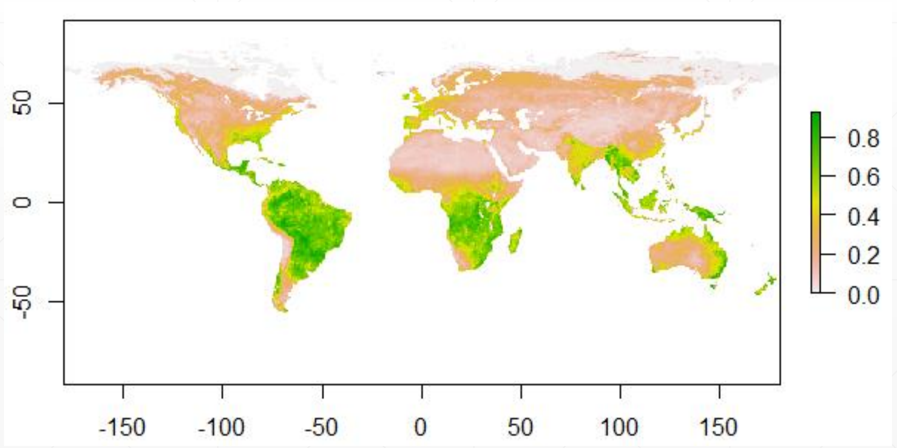
```
ndvi.rb <- brick("GIMMS.NDVI.360.720.2000.2002.30days.nc")
ndvi.rb
ndvi2000 <- ndvi.rb[[1:12]] # select bands as a new raster
plot(ndvi.rb)       # plot all bands
plot(ndvi.rb,6)     # plot a single band
plot(ndvi.rb,1:12) # plot selected bands
```

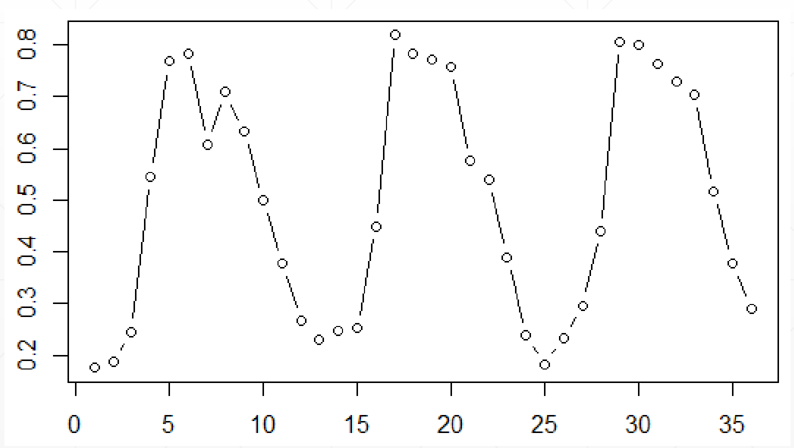


Plot Raster Time Series

- **Get cell values interactively**

```
plot(ndvi.rb,1)
values <- click(ndvi.rb, n=1, xy=TRUE)
values <- click(ndvi.rb, n=1, xy=FALSE)
plot(as.vector(values), type="b")
```

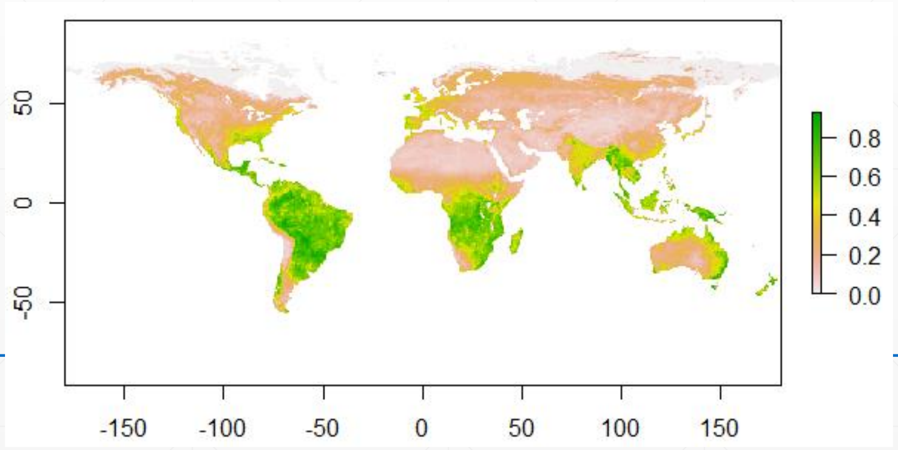


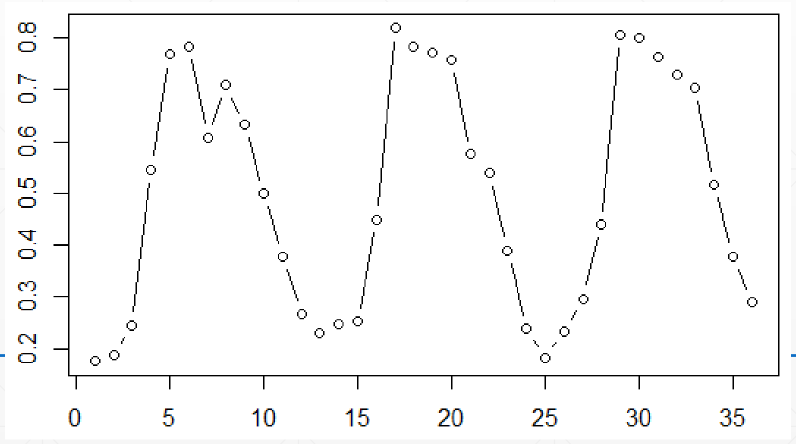


Plot Raster Time Series

- **Get cell values by specifying coordinates**

```
plot(ndvi.rb,1)
cell <- cellFromXY(ndvi.rb, cbind(15, 51))
cell.ts <- ndvi.rb[cell]
cell.ts
plot(as.vector(cell.ts), type="b")
values <- ndvi.rb[50,50]
plot(as.vector(values), type="b")
```


50
0
-50
-150
-100
-50
0
50
100
150
0.8
0.6
0.4
0.2
0.0


0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
5
10
15
20
25
30
35