

AniMove Cheat Sheet

for spatial data handling, statistics and movement analysis

www.animove.org
last updated: April 1, 2014

Packages

raster	for raster data manipulation
sp	for vector data manipulation
rgdal	data import and export, projections
rgeos	geometry commands
spdep	spatial dependence
dismo	species distribution modelling
move	access and analyse movement data
bcpsa	analyse movement tracks

further relevant packages:

spatstat	spatial statistics
gstat	geostatistics
geoR	geostatistical analysis
gdistance	distances on geographical grids
spsurvey	sampling functionality
trip	sp class extension for track analysis
randomForest	random Forest implementation
mgcv	GAM implementation
lme4	mixed-effects model
landsat	Landsat specific analysis
spgrass6	interaction with GRASS

visualisation packages:

maptools	handling spatial objects
maps	map display
mapproj	map projections
mapdata	supplements to maps
rasterVis	enhanced raster visualisation
ggplot2	for more fancy plots

More spatial R packages are listed here:
cran.r-project.org/web/views/Spatial.html

Relevant commands are listed below, actual syntax needs to be checked within the manual pages of each command.

Raster

Raster data manipulation is similar to a spreadsheet or matrix manipulation but with coordinates and projections, hence various also not explicitly spatial commands can be applied. Here we mainly list commands designed for spatial data handling.

Import and export

<code>raster()</code>	import (or generate) one raster layer
<code>brick()</code>	import raster with multiple layers
<code>writeRaster()</code>	export raster data to file
<code>writeFormats()</code>	list of supported raster file types
<code>getData()</code>	retrieves DEM and climate data directly from the web

Information

<code>print()</code>	prints raster metadata
<code>click()</code>	interactively query raster plot
<code>hist()</code>	histogram of raster values per layer
<code>cellStats()</code>	summary statistics of single layers
<code>summary()</code>	summary statistics
<code>extent()</code>	extent of raster data set
<code>ncell()</code>	number of cells (of one layer)
<code>nlayers()</code>	number of bands
<code>names()</code>	prints layer names
<code>str()</code>	print the data structure
<code>NValue()</code>	get or set background values

Visualisation

<code>plot()</code> , <code>plotRGB()</code>	raster plot and RGB plot. Usefull arguments: <code>y=bandnumber</code> , <code>add=TRUE</code> (overlay multiple plots)
<code>image()</code> , <code>spplot()</code>	alternative plotting commands
<code>levelplot()</code>	fancy way to plot raster data information
<code>densityplot()</code>	raster value density plot
<code>bwplot()</code>	violin plot of raster data values
<code>hovmoller()</code>	spatio-temporal plotting options
<code>streamplot()</code>	plotting of streamlines

Projections

<code>projection()</code>	query or set projection (does NOT reproject)
<code>projectRaster()</code>	reprojects raster to new coordinate system

Data manipulation

Most raster commands will output a file to a chosen location, if `filename=` is specified. Otherwise it will use temp files.

<code>stack()</code>	stack different raster layers together
<code>addLayer(); dropLayer()</code>	add/drop a raster layer
<code>crop()</code>	crop raster set to smaller extent
<code>drawExtent()</code>	draw extent on a plot for e.g. inclusion in <code>crop(raster,extent)</code>
<code>mask()</code>	masking of background values
<code>merge(); mosaic()</code>	combine raster tiles to a raster with larger extent
<code>extract()</code>	extract values from Raster objects, using points or polygons
<code>calc()</code>	apply a function to raster data
<code>overlay()</code>	apply a function which uses multiple bands, e.g. to calculate NDVI
<code>focal()</code>	moving window operations
<code>distance()</code>	calculate distance to closest feature, e.g. distance to water
<code>terrain()</code>	calculate terrain attributes from DEM, e.g. slope
<code>zonal()</code>	zonal statistics, for classified raster images, e.g. how many forest pixels
<code>reclassify()</code>	reclassify raster values
<code>subs()</code>	substitutes values
<code>resample()</code>	resampling of raster to another raster
<code>aggregate()</code>	aggregation of cells
<code>disaggregate()</code>	disaggregation of cells
<code>raster*2/raster2</code>	any basic operations can be applied to a raster
<code>[[]]</code>	address specific raster layer, e.g. <code>myRaster[[1]]</code> for first layer of <code>myRaster</code>
<code>x <- raster > 50</code>	boolean operation, output is binary
<code>raster[raster <= 50] <- 0</code>	replace all values smaller then 50 with 0
<code>r1[r1==50] <- r2[r1==50]</code>	values in r1 whose values are equal 50 are replaced by the corresponding values of r2
<code>sampleRandom()</code>	random sample from cell values
<code>sampleRegular()</code>	regular sample from cell values
<code>sampleStratified()</code>	stratified sample from cell values

Vector

Vector data often come in shp format including a variety of auxiliary files. All of them are relevant and are needed for further analysis. Note that `readShapePoly()` etc. from package `maptools` do NOT automatically read projection information from shapefiles. It is recommended to use `readOGR()` instead.

Import and export

<code>readOGR()</code>	import vector file
<code>writeOGR()</code>	export vector file
<code>ogrDrivers()</code>	list supported file formats

Information

<code>plot()</code>	vector plot. <code>add=TRUE</code> overlays multiple plots, e.g. combine with raster data
<code>summary()</code> <code>extent()</code>	metadata and data summary extent/bounding box of vector data
<code>coordinates()</code>	sets spatial coordinates to create spatial data, or retrieves spatial coordinates

Projections

<code>projection()</code>	query or set projection (does NOT reproject)
<code>spTransform()</code>	reproject vector data to new coordinate system

Data manipulation

Check out the functions in the `rgeos` package, which provides most of the classical vector GIS operations such as buffers etc.

<code>subset()</code>	subset spatial data, based on a condition, e.g. keep only certain points
<code>merge()</code>	Merge a Spatial object having a <code>data.frame</code> (i.e. merging of non-spatial attributes)
<code>over()</code>	spatial overlay for points, grids and polygons
<code>rasterize()</code> <code>distanceFromPoints()</code>	Rasterize points, lines, or polygons computes the distance to points, output is a raster
<code>extract()</code>	extracts raster values behind points, lines or polygons
<code>gIntersection()</code> <code>gBuffer()</code>	intersection of vector data sets Buffer Geometry

Spatial Modeling

<code>kfold()</code>	partitioning of data set for training/validation purpose
<code>evaluate()</code>	cross-validation of models with presence/absence data
<code>randomForest()</code>	fits a randomForest model
<code>maxent()</code>	executes Maxent from R
<code>gam()</code>	fits a GAM
<code>pls()</code>	fits a partial least squares model
<code>predict()</code>	predicts statistical model into space (raster)

Movement Analysis

For most of the following commands the data sets need to be converted to a specific format.

<code>move()</code>	import of movement data sets from movebank.org
<code>moveStack()</code>	stacks multiple animal tracks
<code>split()</code>	splits stack into single move objects
<code>movebankLogin()</code>	stores movebank.org credentials
<code>searchMovebankStudies()</code>	reports the studies in movebank.org matching search criteria
<code>getMovebankData()</code>	import tracks directly from movebank.org
<code>show()</code> <code>as()</code>	summary of the move object coerce movement between object types
<code>angle()</code>	extracts turning angles from a move object
<code>speed()</code> <code>distance()</code>	extracts speed from a move object extracts distance between locations from a move object
<code>time.lag()</code>	extracts time lag between locations from a move object
<code>spTransform()</code>	changes the projection of a move object to a default of Azimuthal Equal-distance
<code>mcp()</code>	calculates minimum convex polygons for SpPdf
<code>kernelUD()</code>	calculates a kernel density surface for SpPdf
<code>brownian.bridge()</code>	calculates constant variance Brownian bridges
<code>brownian.bridge.dyn()</code>	calculates dynamic Brownian bridges
<code>LoCoH.k()</code>	calculates local convex hulls using k neighbours
<code>LoCoH.r()</code>	calculates local convex hulls using a radius of r
<code>LoCoH.a()</code>	calculates local convex hulls using an adaptive radius

Miscellaneous

Some useful commands which are related to spatial data analysis.

<code>gmap()</code> <code>geocode()</code> <code>ggplot()</code> <code>ppp()</code> <code>complete.cases()</code>	get google maps for your plot geocoding in R lots of very fancy plotting options creates a point pattern returns only cases with no missing values
<code>gridSample()</code>	sample point from a grid e.g. just one point per pixel
<code>function(...) {...}</code> <code>return(...)</code> <code>if (...) {...} else {...}</code> <code>for (...) {...}</code> <code>while (...) { ...}</code>	generates a defined functions returns the output of a function if else statement for loop while statement

compiled by: Wegmann, Leutner, Bevanda, Horning & Safi
March 2014
<http://www.animove.org>

