



AniMove Cheat Sheet

for animal movement analysis, spatial data handling, remote sensing, spatial statistics and visualization

www.animove.org

last updated: 4th October, 2017

Packages

move	access and analyse movement data
bcpa	analyse movement tracks
ctmm	calculate continuous time movement models
adehabitatHR	home range calculations including classical methods
dismo	species distribution modelling
raster	for raster data manipulation
sp	for vector data manipulation
rgdal	data import and export, projections
rgeos	geometry commands
spdep	spatial dependence

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

visualization packages:

maptools	handling spatial objects
maps	map display
mapproj	map projections
mapdata	supplements to maps
rasterVis	enhanced raster visualization
ggplot2	for more fancy plots
ggmap	map backgrounds for ggplot2
reshape2	flexibly reshape data
moveVis	animating movement and environ. data

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

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

Information

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

Visualization

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

Projections

projection()	query or set projection (does NOT reproject)
raster::projectRaster()	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.

raster::stack()	stack different raster layers together
raster::addLayer()	add/drop a raster layer
raster::dropLayer()	
raster::crop()	crop raster set to smaller extent
raster::drawExtent()	draw extent on a plot for e.g. inclusion in crop(raster,extent)
raster::mask()	masking of background values
raster::merge(); mosaic()	combine raster tiles to a raster with larger extent
raster::extract()	extract values from Raster objects, using points or polygons
raster*2/raster2	any basic operation, more efficient:
raster::calc()	apply a function to raster data and
raster::overlay()	apply a function which uses multiple bands, e.g. to calculate NDVI
raster::focal()	moving window operations
raster::distance()	calculate distance to closest feature, e.g. distance to water
raster::terrain()	calculate terrain attributes from DEM, e.g. slope
raster::zonal()	zonal statistics, for classified raster
raster::reclassify()	reclassify raster values
raster::subs()	substitutes values
raster::resample()	resampling of raster to raster
raster::aggregate()	aggregation of cells
raster::disaggregate()	disaggregation of cells
raster::rasterToPoints()	converts a raster to vector points
raster::rasterToPolygons()	converts a raster to polygons
raster::rasterToContour()	converts raster values to contour
[[]]	address specific raster layer, e.g. myRaster[[1]] for first layer of myRaster
x <- raster > 50	boolean operation, output is binary
raster[raster <= 50] <- 0	replace all values smaller then 50 with 0
r1[r1==50] <- r2[r1==50]	values in r1 whose values are equal 50 are replaced by the corresponding values of r2
raster::sampleRandom()	random sample from cell values
raster::sampleRegular()	regular sample from cell values
raster::sampleStratified()	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>rgdal::readOGR()</code>	import vector file
<code>rgdal::writeOGR()</code>	export vector file
<code>rgdal::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 metadata and data summary
<code>summary()</code>	extent/bounding box of vector data
<code>raster::extent()</code>	sets spatial coordinates to create spatial data, or retrieves spatial coordinates
<code>sp::coordinates()</code>	

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>sp::over()</code>	spatial overlay for points, grids and polygons
<code>raster::rasterize()</code>	Rasterize points, lines, or polygons
<code>raster::distanceFromPoints()</code>	computes the distance to points, output is a raster
<code>raster::extract()</code>	extracts raster values behind points, lines or polygons
<code>rgeos::gIntersection()</code>	intersection of vector data sets
<code>rgeos::gBuffer()</code>	Buffer Geometry
<code>maptools::elide()</code>	Rotate, scale or shift spatial objects

Spatial Modeling

<code>dismo::kfold()</code>	partitioning of data set for training/validation purpose
<code>evaluate()</code>	cross-validation of models with presence/absence data
<code>randomForest::randomForest()</code>	fits a randomForest model
<code>maxent()</code>	executes Maxent from R
<code>mgcv::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. The formats for the `move` packages are based on the `raster` and `sp` and can thus be manipulated using the same functions.

<code>move::move()</code>	import of movement data sets from movebank.org csv's or from loaded data
<code>move::n.locs()</code>	return the number of locations
<code>move::timestamps()</code>	extract timestamps from move objects
<code>move::unusedRecords()</code>	returns the unused records (outliers, non location sensor data, etc)
<code>move::burst()</code>	assign categories to segments for segmented analysis
<code>move::moveStack()</code>	stacks multiple animal tracks
<code>move::UDStack()</code>	stack a list of UD's, convert a RasterStack to UDStack or convert a BurstStack to a UDStack by standardizing.
<code>move::split()</code>	splits movestack into single move objects, or splits a UDStack
<code>move::movebankLogin()</code>	stores movebank.org credentials
<code>move::searchMovebankStudies()</code>	search for a study in Movebank by keywords
<code>move::getMovebankData()</code>	import tracks directly from movebank.org
<code>move::as.data.frame()</code>	create data frame of a move object
<code>move::angle()</code>	calculate headings from a move object
<code>move::turnAngleGc()</code>	calculate turning angles
<code>move::speed()</code>	extracts speed from a move object
<code>move::distance()</code>	extracts distance between locations from a move object
<code>move::timeLag()</code>	extracts time lag between locations from a move object
<code>move::spTransform()</code>	change projection of a move object
<code>move::emd()</code>	calculate differences between UD's or UDStacks
<code>move::raster2contour()</code>	calculate UD contour lines
<code>move::getVolumeUD()</code>	convert UD to UD quantiles
<code>move::interpolateTime()</code>	linearly interpolate locations to specific times to for example regularize a track
<code>move::coordinates()</code>	extract coordinates of a move object
<code>move::getDataRepositoryData()</code>	download data directly from the Movebank Data Repository
<code>move::getDuplicatedTimestamps()</code>	get all pairs of duplicated timestamps
<code>move::getMovebankNonLocationData()</code>	downloads the non location data directly from movebank.org
<code>move::brownian.bridge.dyn()</code>	calculate the utilization distribution (UD) using the dynamic Brownian Bridge Movement Model
<code>move::dynBGB()</code>	calculate the utilization distribution (UD) using the Bivariate Gaussian Bridge model

<code>adehabitatHR::mcp()</code>	calculates minimum convex polygons for SpPdf
<code>adehabitatHR::kernelUD()</code>	calculates a kernel density surface for SpPdf
<code>adehabitatHR::LoCoH.k()</code>	calculates local convex hulls using k neighbours
<code>adehabitatHR::LoCoH.r()</code>	calculates local convex hulls using a radius of r
<code>adehabitatHR::LoCoH.a()</code>	calculates local convex hulls using an adaptive radius

Movement Visualization

Commands to visualize movement, combining both raster and vector data types for displaying animal-environment interactions.

<code>get_imconvert()</code>	Checks, downloads and/or installs ImageMagick needed to create GIF files
<code>animate_move()</code>	animating of movement and environmental data
<code>animate_stats()</code>	animating of movement and environmental data statistics

Miscellaneous

Some useful commands which are related to spatial data analysis.

<code>gmap()</code>	get google maps for your plot
<code>geocode()</code>	geocoding in R
<code>ggplot2::ggplot()</code>	lots of very fancy plotting options
<code>ppp()</code>	creates a point pattern
<code>complete.cases()</code>	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>	generates a defined functions
<code>return(...)</code>	returns the output of a function
<code>if (...) {...} else{...}</code>	if else statement
<code>for (...) {...}</code>	for loop
<code>while (...) { ...}</code>	while statement

compiled by:
Wegmann, Leutner, Bevanda, Kranstauber, Horning, Safi, Schwalb-Willmann & Scharf, 2017
<http://www.animove.org>