AniMove Cheat Sheet

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

www.animove.org
last updated: 27th May, 2019

Packages

`move`	access and analyse movement data
`bcpa`	analyse movement tracks
`ctmm`	continuous time movement models
`recurse`	analyze recursions in movement data
`adehabitatHR`	home range calculations including clasical methods
`dismo`	species distribution modelling
`amt`	step selection function
`raster`	for raster data manipulation
`sp/sf`	for vector data manipulation
`rgdal`	data import and export, projections
`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 explictly 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();
raster::dropLayer()
	add/drop a raster layer

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 reccomended to use readOGR() instead.

Import and export

rgdal::readOGR()	import vector file
rgdal::writeOGR()	export vector file
rgdal::ogrDrivers()	list supported file formats

Information

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

Projections

projection()	query or set projection (does NOT reproject)
spTransform()	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.

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

Spatial Modeling

dismo::kfold()	partitioning of data set for training/validation purpose
evaluate()	cross-validation of models with presence/absence data
randomForest::randomForest()	fits a randomForest model
maxent()	executes Maxent from R
mgcv::gam()	fits a GAM
pls()	fits a partial least squares model
predict()	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.

move::move()	import of movement data sets from movebank.org csv's or from loaded data
move::namesIndiv()	return the names of the individuals
move::n.locs()	return the number of locations
move::angle()	calculate headings
move::turnAngleGc()	calculate turning angles
move::speed()	calculate speed between location
move::distance()	calculate distance between locations
move::timeLag()	calculate time lag between locations
move::spTransform()	change projection of a move object
move::timestamps()	extract timestamps from move objects
move::coordinates()	extract coordinates of a move object
move::unUsedRecords()	extract unused records (outliers, non location sensor data, etc)
move::getDuplicatedTimestamps()	returns all pairs of duplicated timestamps
move::moveStack()	stacks multiple animal tracks
move::split()	splits movestack into single move objects, or splits a UDStack
move::as.data.frame()	create data frame of a move object
move::UDStack()	stack a list of UDs, convert a RasterStack to UDStack or convert a BurstStack to a UDStack by standardizing.
move::burst()	segment track by a specified variable
move::interpolateTime()	linearly interpolate locations to specific times to for example regularize a track
move::thinTrackTime()	select segments from the original track with a fixed time interval
move::brownian.bridge.dyn()	calculate the utilization distribution (UD) using the dynamic Brownian Bridge Movement Model
move::dynBGB()	calculate the utilization distribution (UD) using the Bivariate Gaussian Bridge model
move::raster2contour()	calculate UD contour lines
move::emd()	calculate differences between UDs or UDStacks
move::movebankLogin()	stores movebank.org credentials
move::searchMovebankStudies()	search for a study in Movebank by keywords
move::getMovebankData()	import tracks directly from movebank.org
move::getMovebankNonLocationData()	downloads the non location data directly from movebank.org
move::getMovebankReferenceTable()	downloads the study's reference table from movebank.org
move::getDataRepositoryData()	download data directly from the Movebank Data Repository
adehabitatHR::mcp()	calculates minimum convex polygons for SpPdf
adehabitatHR::kernelUD()	calculates a kernel density surface for SpPdf
adehabitatHR::LoCoH.k()	calculates local convex hulls using k neighbours
adehabitatHR::LoCoH.r()	calculates local convex hulls using a radius of r
adehabitatHR::LoCoH.a()	calculates local convex hulls using an adaptive radius

Movement Visualization

Visualize movement and environmental data using `moveVis`.

view_spatial()	displays tracks on an interactive map
align_move()	aligns tracks to a uniform time scale
frames_spatial()	creates a list of `ggplot2` map frames
frames_spatial()	creates a list of `ggplot2` graph frames
add_gg()	adds `ggplot2` funcions to frames
add_labels()	adds labels to frames
add_scalebar()	adds scalebar to frames
add_northarrow()	adds north arrow to frames
add_progress()	adds progress bar to frames
add_timestamps()	adds timestamps to frames
add_colourscale()	adds colour scale to frames
add_text()	adds text to frames
join_frames()	joins multiple frames side-by-side
animate_frames()	renders frames as GIF or video

Recursion

Compute revisitation metrics for trajectory data with the `recurse` package. Data can be in a `move` object or data frame.

getRecursions()	calculates revisits for every location
getRecursionsAtLocations()	calculates revisits for specified locations
calculateIntervalResidenceTime()	calculates the residence time during userspecified intervals
getRecursionsInPolygon()	calculates revists inside userspecified polygon

Miscellaneous

Some useful commands which are related to spatial data analysis.

geocode()	geocoding in R
ppp()	creates a point pattern
complete.cases()	returns only cases with no missing values
gridSample()	sample points from a grid

compiled by: Wegmann, Leutner, Bevanda, Kranstauber, Horning, Safi, Schwalb-Willmann, Scharf & Bracis, 2019, http://www.animove.org