



## AniMove Cheat Sheet

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

[www.animove.org](http://www.animove.org)

last updated: 20<sup>th</sup> May, 2019

### Packages

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

further relevant packages:

<b>spatstat</b>	spatial statistics
<b>gstat</b>	geostatistics
<b>geoR</b>	geostatistical analysis
<b>gdistance</b>	distances on geographical grids
<b>spsurvey</b>	sampling functionality
<b>trip</b>	sp class extension for track analysis
<b>randomForest</b>	random Forest implementation
<b>mgcv</b>	GAM implementation
<b>lme4</b>	mixed-effects model

visualization packages:

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

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

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

#### Information

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

#### Visualization

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

#### Projections

<b>projection()</b>	query or set projection (does NOT reproject)
<b>raster::projectRaster()</b>	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.

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

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::n.locs()	return the number of locations
move::timestamps()	extract timestamps from move objects
move::unusedRecords()	returns the unused records (outliers, non location sensor data, etc)
move::burst()	assign categories to segments for segmented analysis
move::moveStack()	stacks multiple animal tracks
move::UDStack()	stack a list of UD's, convert a RasterStack to UDStack or convert a BurstStack to a UDStack by standardizing.
move::split()	splits movestack into single move objects, or splits a UDStack
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::as.data.frame()	create data frame of a move object
move::angle()	calculate headings from a move object
move::turnAngleGc()	calculate turning angles
move::speed()	extracts speed from a move object
move::distance()	extracts distance between locations from a move object
move::timeLag()	extracts time lag between locations from a move object
move::spTransform()	change projection of a move object
move::emd()	calculate differences between UD's or UDStacks
move::raster2contour()	calculate UD contour lines
move::getVolumeUD()	convert UD to UD quantiles
move::interpolateTime()	linearly interpolate locations to specific times to for example regularize a track
move::coordinates()	extract coordinates of a move object
move::getDataRepositoryData()	download data directly from the Movebank Data Repository
move::getDuplicatedTimestamps()	get all pairs of duplicated timestamps
move::getMovebankNonLocationData()	downloads the non location data directly from movebank.org
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

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 <b>ggplot2</b> map frames
frames_spatial()	creates a list of <b>ggplot2</b> graph frames
add_gg()	adds <b>ggplot2</b> 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 bar 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 revisits 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