
ANIMOVE
P1.y=diag(c(0, 0.001, 0.001))
playPar(mov.model=~1, err.model=list(x=~errX, y=~errY),
data=nfsNew, fixPar=c(NA, 1, NA, 1, NA, NA, NA, NA))
<- crwMLE(mov.model=~1, err.model=list(x=~errX, y=~errY),
data=nfsNew, coord=c("longitude", "latitude"),
Time.name="Time", initial.state=initial.drift,
fixPar=c(NA, 1, NA, 1, NA, NA, NA, NA),
control=list(maxit=2000,trace=1, REPORT=10),

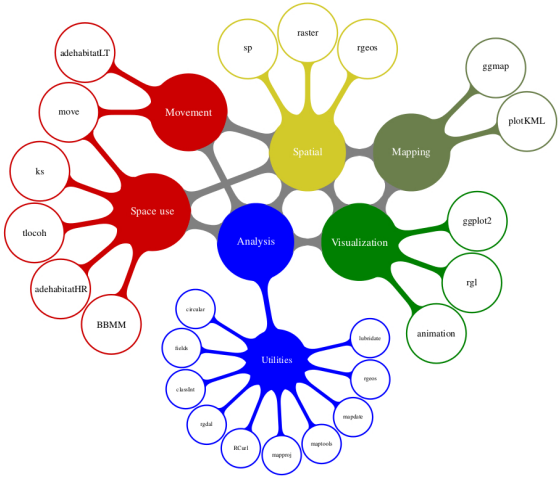
September 2022

Movement data in R

Getting data in and out of R

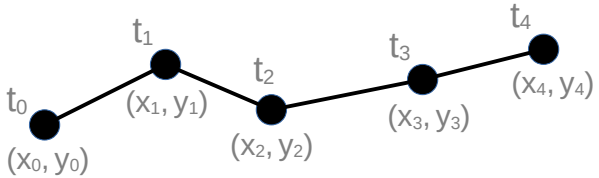
Packages necessary for movement analysis

Because of the properties of movement data, their analysis in R requires the use of a wide range of R packages.



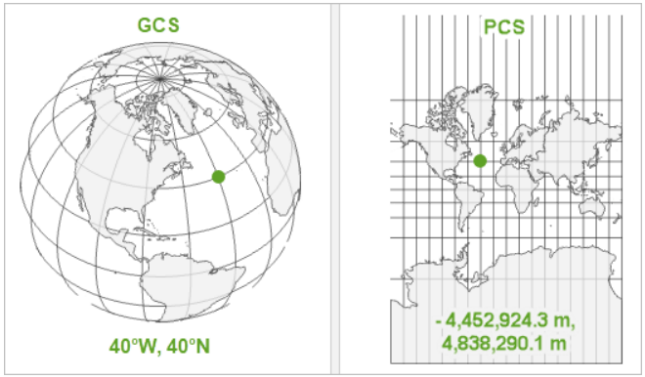
Formal description of movement

Movement is defined as location(s) through time. Movement is a spatio-temporal object. Correct definition of space and time will allow us to manipulate the data in an appropriate way.



Problem 1: Space

The Earth is a sphere, but paper maps and computer screens are flat. A geographic coordinate system (GCS) defines where data are located on the Earth's surface taking into account its curvature (degrees); a projected coordinate system (PCS) tells us how these data can be drawn in 2D (metres).



Problem 1: Space

Try to flatten the skin of a peeled orange on the table without crinkles and cuts; it is not possible, the consequence being: distance, area and angle can **not** be mapped in a 2D coordinate system without some degree of distortion.

But, by correctly defining (declaring) the projection we can minimise the distortion in the measures we are interested in, also depending on the scale.

Helpful webpage: `https://www.esri.com/arcgis-blog/products/arcgis-pro/mapping/gcs_vs_pcs/`
`https://learn.arcgis.com/en/projects/choose-the-right-projection/`

Projection in R: declaration vs. reprojection

In R we can do that by working with proper *spatial objects*. But careful, declaring a projection:

```
> coordinates(YourData) <- ~location.long+location.lat
> proj4string(YourData) <- CRS("+proj=longlat +datum=WGS84") # or
> projection(YourData) <- CRS("+proj=longlat +datum=WGS84")

> projection(YourData)

[1] "+proj=longlat +datum=WGS84"

> class(YourData)

[1] "SpatialPoints"
```

is different from reprojecting, i.e. changing the coordinate system:

```
YourDataProj <- spTransform(YourData, CRS("+proj=UTM +zone=22 +datum=WGS84"))
```

Helpful webpage to find the corresponding proj4 syntax of a projection: `http://spatialreference.org/`

Problem 2: Time

Time settings are crucial. You have to know what your device used as time. Many GPS devices use and report local time, some report UTC!

Helpful questions you have to ask yourself before importing your data are:

- Did you track over the day light saving switches?
- In which time zone does your device report time?
- Did you set the time zone?

Time in R:

Time format in R is not a real problem, but time zones are OS specific (check yours with `Sys.timezone()` and `OlsonNames()`), and they are likely to be a pain in the neck, no matter how experienced you are..

`POSIXct` is our object of choice to deal with time in R:

```
> (t <- as.POSIXct("2022-09-12 14:00:00", "%Y-%m-%d %H:%M:%S", tz="CET"))

[1] "2022-09-12 14:00:00 CEST"
```

Once time zone is set, time can be converted to different time zones:

```
> format(t, tz="UTC") # or:
> lubridate::with_tz(t, tz="UTC")

[1] "2022-09-12 12:00:00 UTC"
```

But if we omit setting the time zone, R will use your local (computer!) time zone, check it with:

```
> Sys.time()
```

Movement in R:

To deal with movement data in R, many different objects exist, however, many of them neglect one or a few important aspects.

The `Move` package extends the spatial object `SpatialPointsDataFrame` by making time a mandatory component (**movement = location + time**).
It also allows to calculate distances taking into account the curvature of the Earth and it makes sure some basic assumptions of movement are not violated (e.g. duplicated times are not allowed).

For a detailed introduction to the package see: `https://bartk.gitlab.io/move/articles/move.html`