A7 Die R-Pakete sp und spdep

Jan-Philipp Kolb

22 Oktober 2018

Themen dieses Abschnitts

- Eine räumliche Stichprobe ziehen
- Adressen für die gezogenen Punkte bestimmen
- Adressdatensatz bereinigen
- Wie lässt sich die Entfernung bestimmen

Das erste Gesetz der Geographie (TFLG)

"All things are related, but nearby things are more related than distant things" [Tobler, 1970]

Shapefile mit Regionalschlüssel herunterladen

```
library(rgdal)
```

```
## Loading required package: sp
```

```
## rgdal: version: 1.3-2, (SVN revision 755)
##  Geospatial Data Abstraction Library extensions to R succes
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: D:/Eigene Dateien/Dokumente/R/w
##  GDAL binary built with GEOS: TRUE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VER
##  Path to PROJ.4 shared files: D:/Eigene Dateien/Dokumente/R
##  Linking to sp version: 1.3-1
```

```
setwd(vg250path)
VG250 <- readOGR ("VG250_GEM.shp","VG250_GEM")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "D:\GESIS\data\vg250_3112_utm32s_shape_ebenen\vg250
```

Räumliche Stichprobe

- Mit der Funktion `spsample` aus dem Paket `sp` kann man eine räumliche Stichprobe ziehen.

```
spatsamp <- spsample(VG250, 100,type="random")
```

`type`	character; `"random"` for completely spatial random; `"regular"` for regular (systematically aligned) sampling; `"stratified"` for stratified random (one single random location in each "cell"); `"nonaligned"` for nonaligned systematic sampling (nx random y coordinates, ny random x coordinates); `"hexagonal"` for sampling on a hexagonal lattice; `"clustered"` for clustered sampling; `"Fibonacci"` for Fibonacci sampling on the sphere (see references).

Point in Polygon

- Mit der Funktion over kann man feststellen in welchem Polygon ein Punkt liegt.

```
tmp <- sp::over(spatsamp, VG250)
```

```
head(tmp)
```

```
##   ADE GF BSG           RS      AGS       SDV_RS       GEN
## 1   6  4   1 120695904052 12069052 120695904052 Borkheide
## 2   6  4   1 147300180180 14730180 147300180180  LÃ¶bnitz
## 3   6  4   1 160755004063 16075063 160755004063    LÃ¶hma
## 4   6  4   1 092785248134 09278134 092785248134 Haselbach
## 5   6  4   1 034600003003 03460003 034600003003  Dinklage
## 6   6  4   1 032410017017 03241017 032410017017   Springe
##                      BEM NBD SN_L SN_R SN_K SN_V1 SN_V2 SN
## 1 gemeinschaftsangehÃ¶rig  ja   12    0   69    59    04
## 2                     --  ja   14    7   30    01    80  1
```

Daten in ein anderes CRS übertragen

```
library(sp)
```

spTransform for map projection and datum transformation

```
# EPSG: 3857
newData<-sp::spTransform(spatsamp, CRS("+init=epsg:3857"))
```

Eine Karte von Afrika

```
library(maptools)
data(wrld_simpl)
Africa <- wrld_simpl[wrld_simpl@data$REGION==2,]
plot(Africa)
```

Das Zentrum eines Polygonzuges

```
Af <- coordinates(Africa)
head(Af)
```

```
##            [,1]         [,2]
## DZA   2.627813   28.1721102
## AGO  17.552463  -12.3503789
## BEN   2.332296    9.6047655
## COG  15.218362   -0.8732659
## COD  23.646032   -2.8711605
## BDI  29.901786   -3.3461606
```

Die Koordinaten plotten

```
plot(Africa)
points(x=Af[1,1],y=Af[1,2],col="red",pch=20)
```

Die nächsten Nachbarn finden

```
library(spdep)
Af_nb <- tri2nb(Af)
```

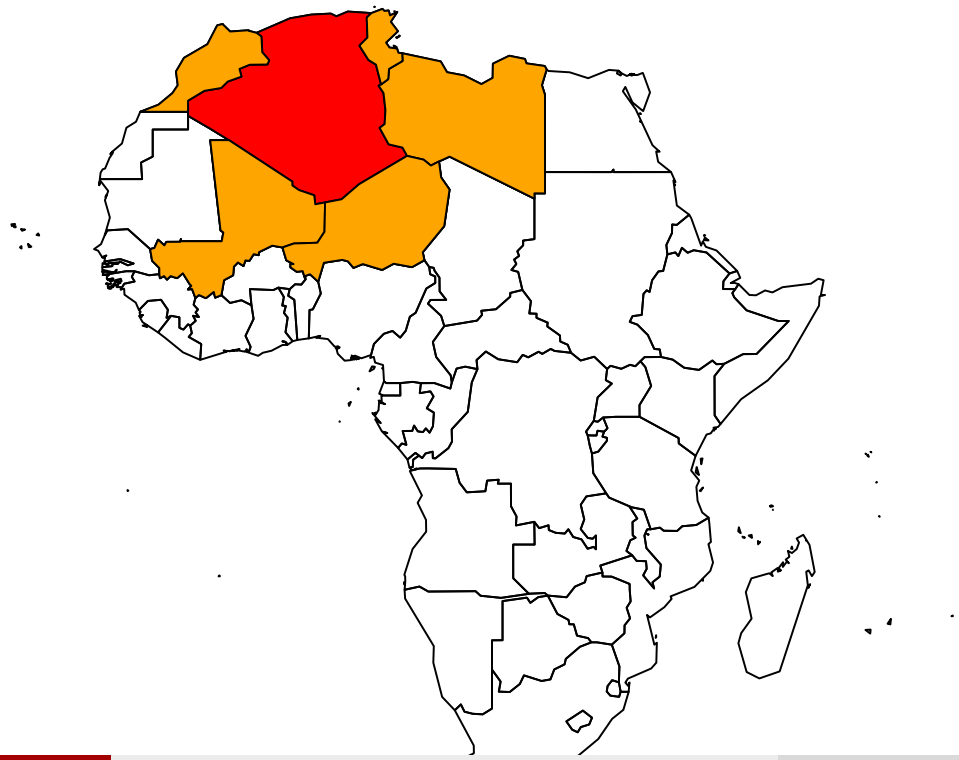
Die Nachbarn für das erste Land:

```
Af_nb[1]
```

```
## [[1]]
## [1] 24 26 27 32 48
```

Die Nachbarn finden

```
plot(Africa)
plot(Africa[1,],col="red",add=T)
plot(Africa[Af_nb[1][[1]],],col="orange",add=T)
```



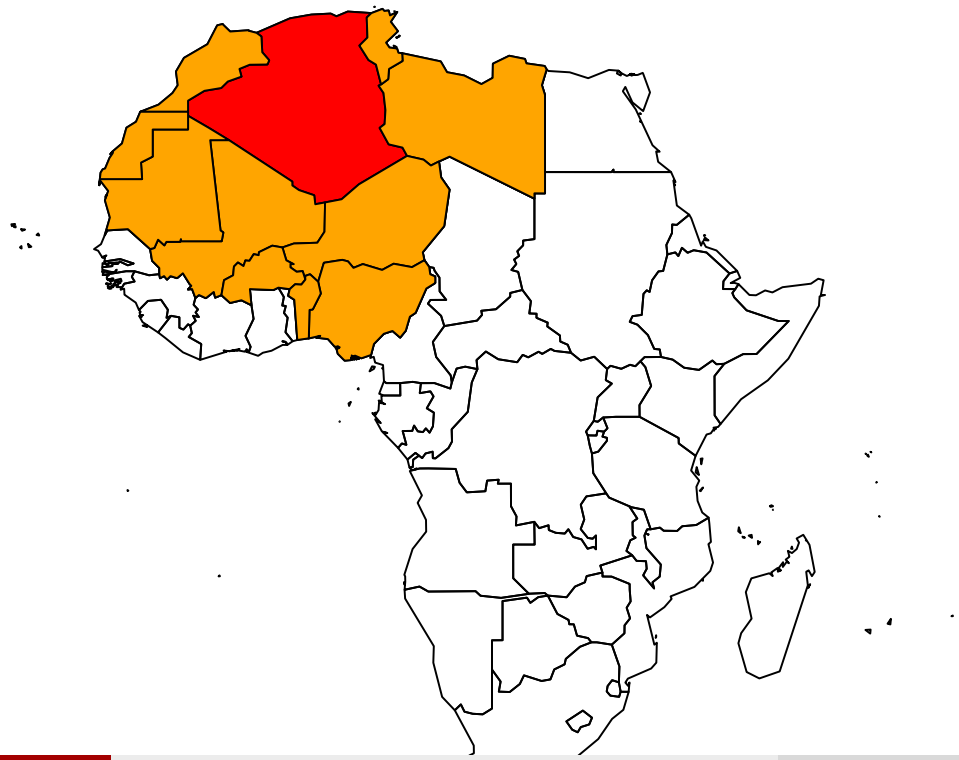
k nearest neighbours

```
IDs <- row.names(as(Africa, "data.frame"))
(Af10_nb <- knn2nb(knearneigh(Af, k = 10), row.names = IDs))
```

```
## Neighbour list object:
## Number of regions: 57
## Number of nonzero links: 570
## Percentage nonzero weights: 17.54386
## Average number of links: 10
## Non-symmetric neighbours list
```

Die 10 nächsten Nachbarn finden

```
plot(Africa)
plot(Africa[1,],col="red",add=T)
plot(Africa[Af10_nb[1][[1]],],col="orange",add=T)
```



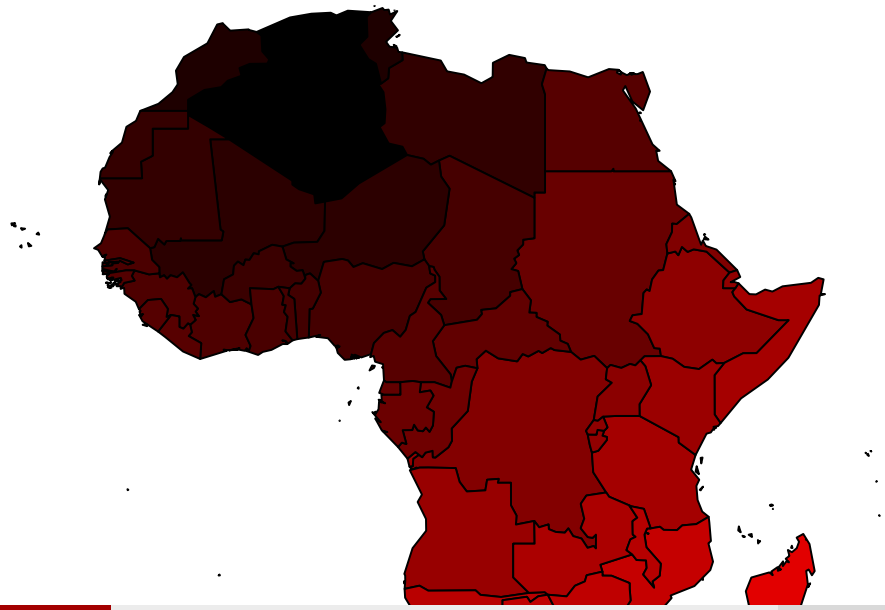
Die Distanz berechnen

```
Af <- coordinates(Africa) # get centroid
library(raster)
pointDistance(Af[1:4,], lonlat=TRUE) # compute distance
```

```
##           [,1]    [,2]    [,3] [,4]
## [1,]         0      NA      NA   NA
## [2,] 4763231.       0      NA   NA
## [3,] 2055609. 2954497.       0   NA
## [4,] 3484053. 1295173. 1839191.    0
```

Berechnen/zeichnen einer Distanzmatrix

```
Dist_Af <- pointDistance(Af, lonlat=TRUE)
Af_color <- Dist_Af[,1]
Af_color <- Af_color/max(Af_color)
Af_color <- rgb(Af_color,0,0)
plot(Africa,col=Af_color)
```



A7A Übung - Nachbarschaften in London

- Lade den Datensatz london_sport von meinem Github Verzeichnis herunter.
- Importiere den Datensatz.
- Bestimme die nächsten Nachbarn des Stadtteils *City of London*

```
setwd("D:/github/geocourse/data/")
london_sport <- readOGR ("london_sport.shp","london_sport")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "D:\github\geocourse\data\london_sport.shp", layer:
## with 33 features
## It has 4 fields
## Integer64 fields read as strings:  Pop_2001
```

Links

- **Raster, CMSAF and solaR**

https://procomun.wordpress.com/2011/06/17/raster-cmsaf-and-solar/

- **Getting rasters into shape from R**

https://johnbaumgartner.wordpress.com/2012/07/26/getting-rasters-into-shape-from-r/