USIC - UNV

JcB

9 juillet 2016

Contents

Distancier des UNV

Objectif: calculer un distancier entre les Villes RGE et les UNV et USIC (voir aussi "RPU_Carto_Pop/cartographie/hop_alsace

1. Récupérer les coordonnées des villes

- source: fichier **geofla 2016 - communes** de l'IGN. Le dossier comprte plusieurs sous dossier. On ne cnserve que le dossier **COMMUNE** qui est du tpe *shapefile*. Ce dossier contient 2 shapefiles COLMMUNE et LIMITES_COMMUNE. On ne conserve que le cdossier COMMUNE sous le nom de **COM**.

```
path <- "../Data/COM"
file <-"COMMUNE"
france <- readOGR(dsn = path, layer = file, encoding = "latin1") # règle le pb de accents

# projection des données
france@proj4string
```

2. Transformer les coordonnées en WSG84

Reprojeter en WGS 84 , code EPSG : 4326

```
france_WGS84 <- spTransform(france, CRS ("+init=epsg:4326"))
```

La transformation fonctionne mais en fait seule les coordonnées des points sont affectées et non le conenu de *data* qui est un dataframe. Il faut donc extraire de *data* les données de géolocalisation et les transformer en *spatialPoints*.

```
d <- france@data[, c("X_CHF_LIEU", "Y_CHF_LIEU", "INSEE_COM")]

# transformation en spatial points
coordinates(d) = ~ X_CHF_LIEU + Y_CHF_LIEU # transformation en sptialPoint
proj4string(d) = france@proj4string # attribue le référentiel d'origine (Lambert 93)

# Transformation en wsg84
d2 <- spTransform(d, CRS ("+init=epsg:4326"))

# coordoonnées des villes en WSG
x <- d2@coords # matrice de 2 colonnes
names(x) <- c("Lon_CHF_LIEU", "Lat_CHF_LIEU")

# Ajout des coord.WSG au SpatialPolygon
france@data <- cbind(france@data, x)
names(france)

# test
france@data[france@data$NOM_COM == "ANDLAU", c("NOM_COM", "Lon_CHF_LIEU", "Lat_CHF_LIEU")]

# Sauvegarde
save(france, file = "sp_France.Rda")
```

communes de la région grand est

rge <- france@data[france@data\protect\T1\textdollarCODE_DEPT %in% c(67, 68, 88, 57, 54, 55, 51, 52, 10, “08”), c(“NOM_COM”, “INSEE_COM”, “Lon_CHF_LIEU”, “Lat_CHF_LIEU”)]

communes d’Alsace

als <- france@data[france@data\protect\T1\textdollarCODE_DEPT %in% c(67, 68), c(“NOM_COM”, “Lon_CHF_LIEU”, “Lat_CHF_LIEU”)]

“‘

3. Création du distancier

```
library(osrm)

# fichier des UNV
unv <- read.csv("../UNV.csv")
# on réordonne les colonnes
unv <- unv[, c(1, 3, 2)]

# boucle pour ne pas dépasser les limites du serveur
# d <- osrmTable(src = als[1:5,], dst = unv) # test
```

```
n <- nrow(als) # nb de villes en Alsace
d1 <- NULL
i <- 1
j <- 1
inc <- 90
k <- inc

while(i < n){
d <- osrmTable(src = als[j:k,], dst = unv)
    d1 <- rbind(d1, d$durations)
    j <- k + 1
    k <- k + inc
    if(k > n)
        k = n
    i = k
}
# une boucle supplémentaire pour les dernières villes
d <- osrmTable(src = als[j:k,], dst = unv)
d1 <- rbind(d1, d$durations)

# sauvegarde
write.csv(d1, file <- "distancier_unv_alsace.csv")
```

UNV la plus proche

Pour une commune, quelle est l'UNV la plus proche ?

```
d1 <- read.csv("distancier_unv_alsace.csv")

d1$choix1 <- NA
for(i in 1:nrow(d1)){d1$choix1[i] = names(which.min(d1[i,2:length(d1)]))}
```

Quel est le meilleur choix si Haguenau n'est pas disponible ?

```
# sans Haguenau
d1$choix2 <- NA
for(i in 1:nrow(d1)){
    d1$choix2[i] = names(which.min(d1[i, c(2, 4:length(d1))]))
    }

head(d1)
```

```
##            X   HTP CH.Haguenau CH.Colmar CH.Mulhouse CH.Belfort CH.Epinal
## 1 DUPPIGHEIM  15.1        34.6      42.0        65.1       83.2     111.1
## 2    LUCELLE 109.7       129.2      71.3        48.9       45.0     110.2
## 3 ARTOLSHEIM  46.4        65.9      29.6        51.4       69.5     103.4
## 4  INNENHEIM  15.8        35.4      37.5        60.6       78.7     106.6
## 5  BUSWILLER  26.7        22.0      73.0        96.1      114.1     117.7
## 6   MULHOUSE  69.8        89.4      31.5         4.7       29.6      92.2
##   Bar.le.Duc CH.Forbach CH.Metz..Mercy. CHU.Nancy..Central.      choix1
## 1      164.6       82.3           103.0               109.9         HTP
## 2      205.1      172.3           186.4               149.1  CH.Belfort
```

```
## 3      168.3      109.0           129.8               113.6   CH.Colmar
## 4      165.4       83.0           103.8               110.7         HTP
## 5      144.1       61.8            82.5                89.4 CH.Haguenau
## 6      187.1      132.5           153.2               131.1 CH.Mulhouse
##        choix2
## 1         HTP
## 2  CH.Belfort
## 3   CH.Colmar
## 4         HTP
## 5         HTP
## 6 CH.Mulhouse
```

Deuxième choix possible si le plus proche n'est pas disponible. Adaptation de la méthode ___http://stackoverflow.com/questions/2453326/fastest-way-to-find-second-third-highest-lowest-value-in-vector-or-column__. *choix* correspond au rang du choix.

```
choix = 2
d1$choix3 <- NA
for(i in 1:nrow(d1)){
    x <- d1[i, 2:11]
    n <- length(x) # en fait c'est une constante
    y = sort(x,partial=n-1)[choix]
    d1$choix3[i] = names(y)
    }
```

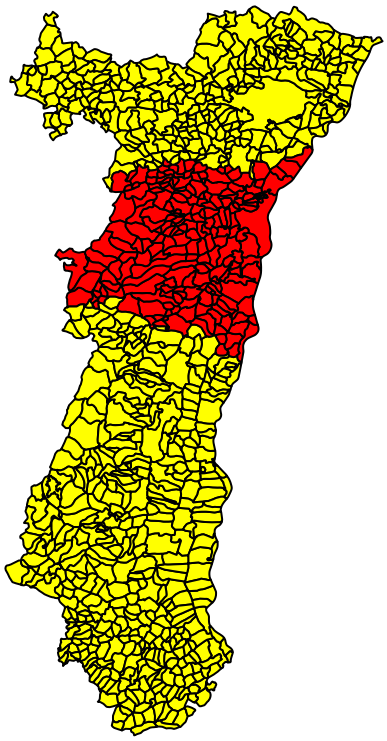
head(d1)

Cartographie

```
#france
load("sp_France.Rda")

alsace <- france[france@data$CODE_DEPT %in% c("67","68"),]
par(mar = c(0,0,2,0))
plot(alsace)
```

```
# alsace et d1 conservent le même prdre des communes, ce qui facilite l'ajout des colonnes choix au dat
alsace@data$CHOIX1 <- d1$choix1
plot(alsace, col = ifelse(alsace$CHOIX1 == "HTP", "red", "yellow"))
```



```
alsace@data$CHOIX2 <- d1$choix2
plot(alsace, col = as.numeric(as.factor(alsace$CHOIX2)) + 1, border = "gray")
```