Using R as a Geographical Information System

F. Vanwindekens

Gembloux – June 2016

Contents

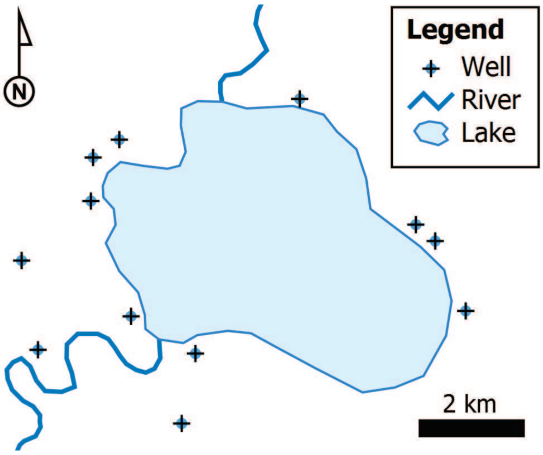
Contents

1 Shapefiles

1.1 Definitions

- *geospatial vector* data format (ESRI, 90's)
- now, *open standard* for spatial data interoperability
- vector features: *points, lines, and polygons.*

source : wikipedia



source : wikipedia

1.2 Loading data

Packages

```
library(sp) # vector data
library(rgdal) # important gis library, input/ouput, projection
```

Functions

```
readOGR(dsn="FOLER",
        layer="FILE",
        input_field_name_encoding="latin1"
        )
```

ex. : Polygons

```
provinces <- readOGR(dsn="./data/AdminVector_2011_L72_shp",layer="AD_4_Province",input_field_name_encoding="latin1")

summary(provinces)
```

ex. : Points and lines

```
city <- readOGR(dsn="./data/City",layer="city")

roads <- readOGR(dsn="./data/Roads",layer="RoadMap")
```

1.3 Data

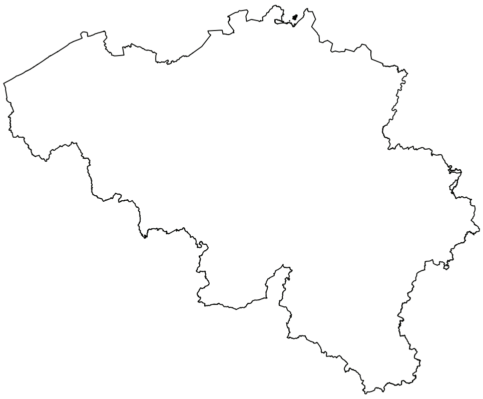
Attribute data

```
names(provinces) # names of variables

provinces@data # data frame of the attrubute data

dim(provinces@data) # dimension of the data frame
head(provinces@data) # dimension of the data frame
```

1.4 Mapping

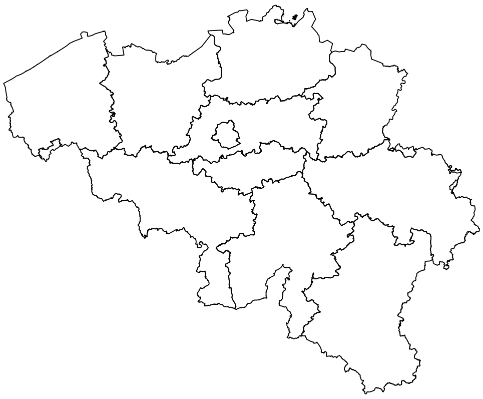
Basic maps : Belgium

```r
plot(belgium)
```



Basic maps : provinces

```r
plot(provinces)
```

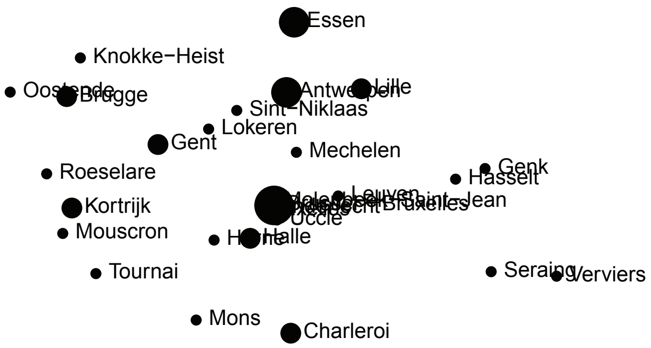


Shapes attributes based on data : color, cex, lwd, . . .

```
# one color for each province
plot(provinces,col=provinces$NISCODE)

# one color for each type of roads
plot(roads,
     lwd=roads@data$FUNCRDCL,
     col=roads@data$FUNCRDCL)

# size of the point varying according to the level of the city
plot(city,cex=(14-(city$LEVEL)),pch=19)
```

Labelling using "text"

```
plot(city,cex=(14-(city$LEVEL)),pch=19)
text(city[!is.na(city$LEVEL),],labels=city$NAME[!is.na(city$LEVEL)
    ],pos='4')
```



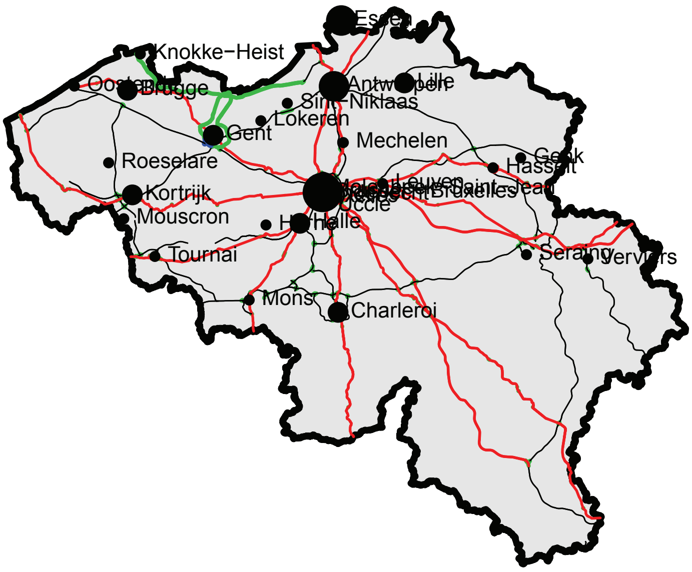
Combining different shapes using "add=TRUE"

```r
# only one polygon in green
plot(provinces) # plot all provinces
plot(provinces[provinces$NISCODE==90000,],col="green",add=TRUE)
```

Building a first whole map

```r
plot(belgium,lwd=5,col=grey(0.9))

plot(roads,lwd=roads@data$FUNCRDCL,col=roads@data$FUNCRDCL,add=TRUE
    )

plot(city,cex=(14-(city$LEVEL)),pch=19,add=TRUE)
text(city[!is.na(city$LEVEL),],city$NAME[!is.na(city$LEVEL)],pos='4
    ')
title("Major cities and roads of Belgium")
```

Major cities and roads of Belgium



Mapping with ggplot2

Packages

```
library(ggplot2) # vizualisation
library(plyr) # for join, better than merge
library(grid) # for function unit in theme of ggplot
```

- ggplot2 does not deal with spatial objects ’out-of-the-box’
- but it deals with data frames

Mapping with ggplot2

Functions for formatting data

```
# create an 'id' if not present
SHAPEFILE@data$id <- ID

# get the coordinates of all spatial objects of the shapefile
DATA.FRAME <- fortify(SHAPEFILE)

# link the attribute data to the new data frame
DATA.FRAME.2 <- join(DATA.FRAME,SHAPEFILE@data,by="id")
```

Most important variables of the new data frame

- `long` – x coodinates
- `lat` – y coordinates
- `group` – *identifies the groups of coordinates that pertain to individual polygons*

Basic functions for visualising data

```
MAP <- (ggplot(DATA.FRAME, aes(long,lat)))

MAP.2 <- (MAP
          + geom_polygon(aes(group=group),fill='COLOR')
          )

print(MAP.2)
```

Basic geom and options

- `+ geom_path(...)` – lines, borders of polygons
- `+ geom_point(...)` – points
- `+ coord_equal()` – fix the ratio between x and y coordinates
- `+ labs` – labelling the maps

2 Manipulating data

2.1 Formatting data

Get the data on the web

- `http://www.atlas-belgique.be/cms2/#`
- select a criteria > see the map
- Near the scale > 'i' > Export values

Excel → .csv/.txt/...

- check, remove lines (?), ...
- NA for "not available" data
- 'save as' > .txt (sep. tabulation) | .csv (sep ";")

Loading the data

```
DATA.FRAME <-
   read.delim2("./FOLDER/FILE.txt",    # your file
               dec=",",         # decimal character
               stringsAsFactors=FALSE)  # explicit
```

Formatting the data

```
names(DATA.FRAME)[1] <- AN.ID
# duplicate the major shapefile
NEW.SP.OBJ <- SHAPEFILE
# add the attribute from
NEW.SP.OBJ@data <- join(SHAPEFILE@data,
                        DATA.FRAME,
                        by='AN.ID') # use names() <-
```

Example

- 'Part des exploitations avec des légumes en plein air'
- Basic map colored according these values

2.2 Mapping+

Plotting the data using ggplot2

```
# pre-treatment
fortify(...)
join(...)

# mapping
map <- (ggplot(DATA.FRAME, aes(long,lat))
  + geom_polygon(aes(group=group,fill=valeur))
  + coord_equal()
)
print(map)
```

Defining a new theme : theme.map.white

```
theme.map.white <- theme(
    plot.title=element_text(size=rel(1.5),hjust=0),
    # panel.grid.minor = element_blank(),
    # panel.grid.major = element_blank(),
    panel.background=element_blank(),
    plot.background = element_blank(),# element_rect(fill="#e6e8ed
        "),
    panel.border = element_blank(),
    axis.line = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    legend.position="bottom",
    legend.direction="horizontal",
    legend.key.width=unit(0.1,'npc')
    )
```

Creating map.admin – a list of spatial layers

```
map.admin <- list(
    geom_polygon(data=belgium,
                 aes(x=long, y=lat, group=group),
                 fill=NA,color="grey50", size=1),
    geom_polygon(data=provinces,
                 aes(x=long, y=lat, group=group),
                 fill=NA,color="grey50",size=0.5)
    )
```

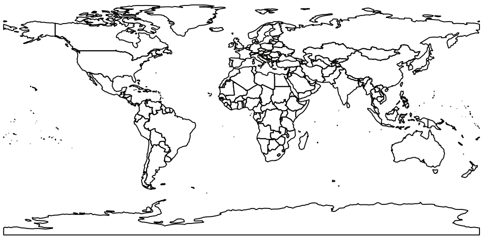
'belgium' and 'provinces' are *spatial objects*
no need to fortify these layers ! (why???)

see also : `map.info`
a personal scale bar and a north arrow for Lambert Belge 72

2.3 Projection

Ex. Worl Map

```
library(rworldmap)
w <- getMap()
plot(w)
```



Get the projection - proj4string

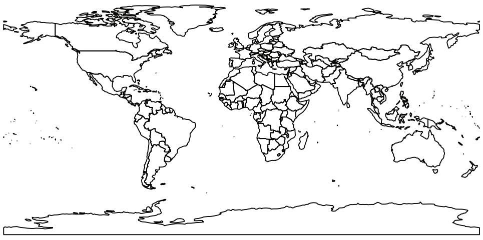
```
summary(w)
```

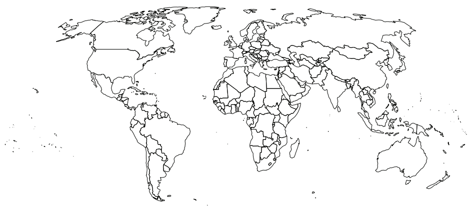
```
Object of class SpatialPolygonsDataFrame
Coordinates:
   min       max
x -180 180.00000
y  -90  83.64513
Is projected: FALSE
proj4string :
[+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs]
Data attributes: [...]
---
# get the proj4string string :
proj4string(w)
```

List of all projections : *http://spatialreference.org/*

Change the projection : Robin

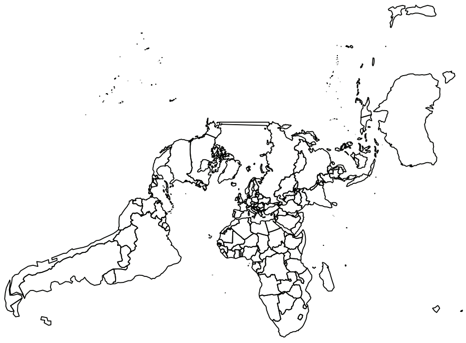
```
w.robin <-
    spTransform(w,CRS("+proj=robin +ellps=WGS84"))
plot(w.robin)
```





Change the projection : Lambert belge 72

```
w.lb72 <- spTransform(w,"+proj=lcc +lat_1=51.16666723333333 +lat_2=49.8333339 +lat_0=90 +lon_0=4.367486666666666 +x_0=150000.013 +y_0=5400088.438 +ellps=intl +towgs84=106.869,-52.2978,103.724,-0.33657,0.456955,-1.84218,1 +units=m +no_defs")
plot(w.lb72)
```



lien détente : `https://www.nfb.ca/film/impossible_map`

2.4 Geostatistics

Formatting data

```
DATE.FRAME <- read.table(...)

# transform to a spatial object
coordinates(DATA.FRAME) <- ~long+lat

# attribute a projection
proj4string(SP) <- CRS("+init=epsg:4326")
## ie "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"

# change the projection
SP.2 <- spTransform(SP,crs(ANOTHER_SP))
```

Spatial analysis

```
# select only some points
selection <- over(SP.POINTS,SP.POLYGONS)
SP.POINTS.2 <- SP_POINTS[!is.na(selection[,1]),]

# analysis, e.g. count the points per polygon
```

```
SP.POINT.POLYGONS <- aggregate(x=SP.POINTS.2,by=SP.POLYGONS,FUN=
    length)
```

Mapping

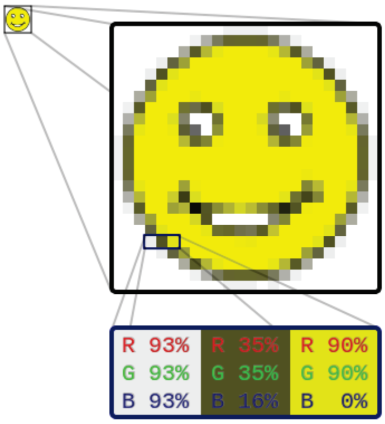
...

3 Rasters

3.1 Definitions

dot matrix data structure representing a generally rectangular grid of pixels, or points of color

(e.g. remote sensing data)

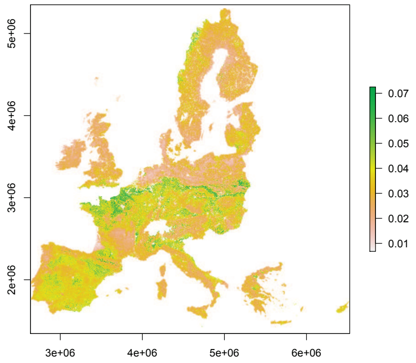


source : wikipedia

3.2 Data

Package & basic loading function

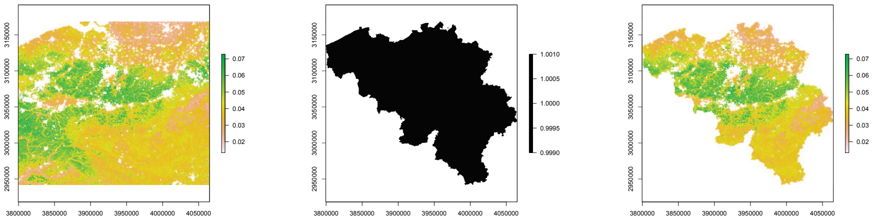
```
library(raster) # raster data
MAP <- raster(``./FOLDER/FILE.tif'')
```



Crop – Frame – Clip

```
# cropping according to the bounding box
CROPPED.RASTER.MAP <- crop(RASTER.MAP,OTHER_MAP)

# framing : rasterizing an SP object according to the tiles of a
    raster
FRAME.RASTER.LAP <-
    rasterize(SP.POLYGONS,CROPPED.RASTER.MAP)

# clipping
CLIPPED.RASTER.MAP <-
    mask(x=CROPPED.RASTER.MAP,mask=FRAME.RASTER.MAP)
```

Crop – Frame – Clip



Function for changing projection, resolution, . . .

```
projectRaster(from=..., to=...,res=...,crs=...)
```

Tip : defining a raster.base

```
raster.base <-
    projectRaster(A.GOOD.RASTER,res=YOUR_RESOLUTION,crs=crs(belgium))
```

This will be useful for defining the same structure (exent, resolution, projection,...) for all your rasters

```
NEW.RASTER <- projectRaster(RASTER,raster.base,crs=crs(belgium))
```

Saving your data

```
writeRaster(YOUR.RASTER,
            filename="./FOLDER/FILE.tif",
            format="GTiff",
            overwrite=TRUE)

## can be read by raster("./FOLDER/FILE.tif")
```

3.3 Mapping

basic plot

```
plot(RASTER)
```

with ggplot2

```
DATA.FRAME <- as.data.frame(RASTER,xy=TRUE,na.rm=TRUE)

map <- (ggplot(aes(x = x, y = y),data=DATA.FRAME)
               + geom_tile(aes(fill=CRITERIA))
               + labs(title="TITRE")
               + scale_fill_gradientn()
               + coord_equal()
               ## + map.info
               ## + map.admin
               ## + theme.map.white
               )
```

3.4 Analysis

Some functions (package raster)

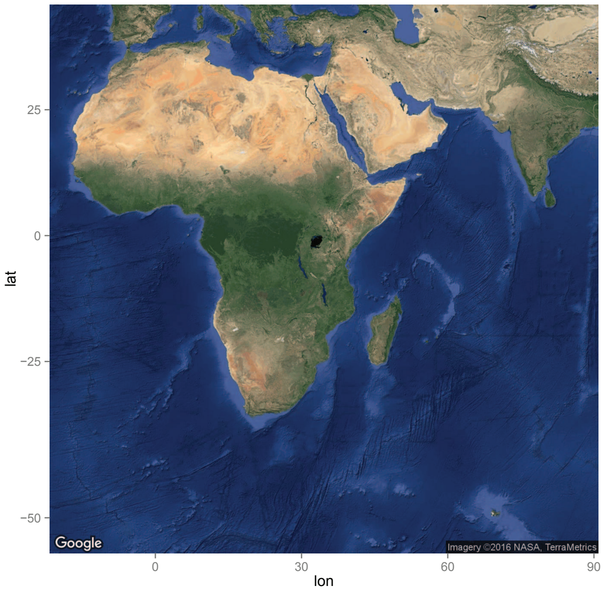
```
alt.be <- getData('alt', country='BEL',path='./data/getdata/')

slope.be <- terrain(alt.be, opt='slope')

aspect.be <- terrain(alt.be, opt='aspect')

hill.be <- hillShade(slope.be, aspect.be, 40, 270)
```

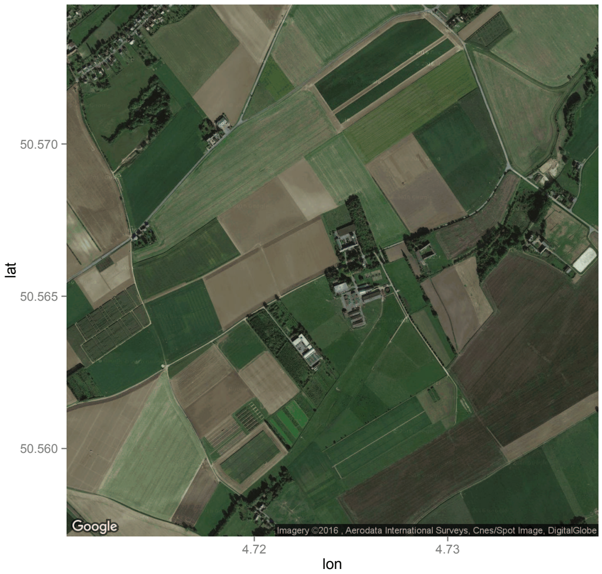
ex. Establishing vineyards in Belgium

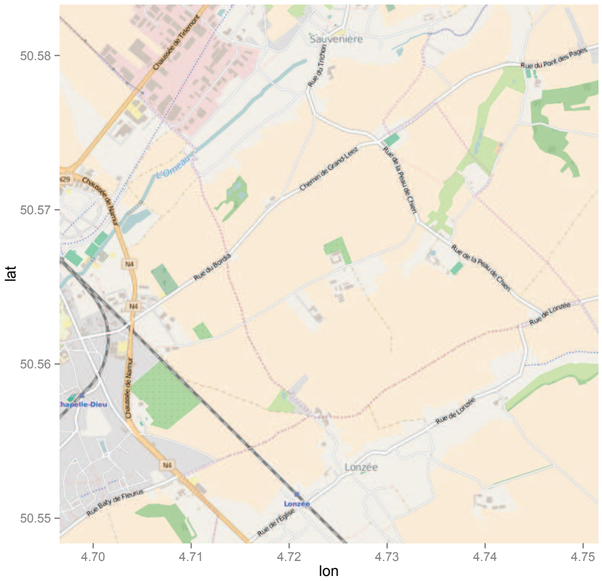
- 'Nul'
- (Slope $>$ 2%) & (W $<$ Aspect $<$ E) → 'Acceptable'
- (Slope $>$ 2%) & (SW $<$ Aspect $<$ SE) → 'Bon'
- (Slope $>$ 4%) & (SW $<$ Aspect $<$ SE) → 'Supérieur'

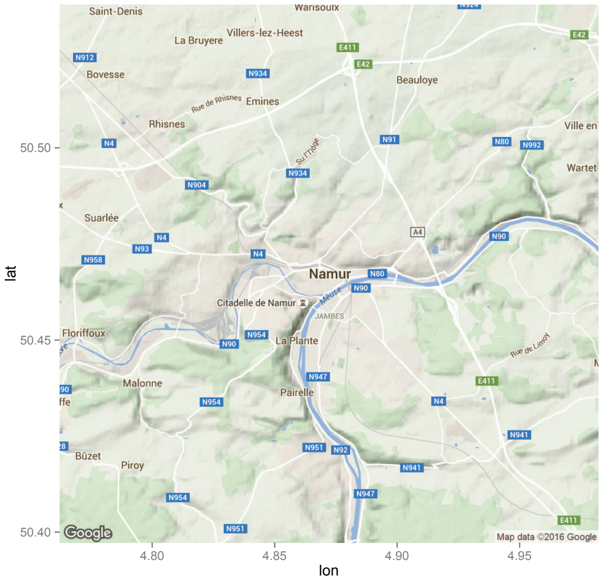
4 Artistic maps

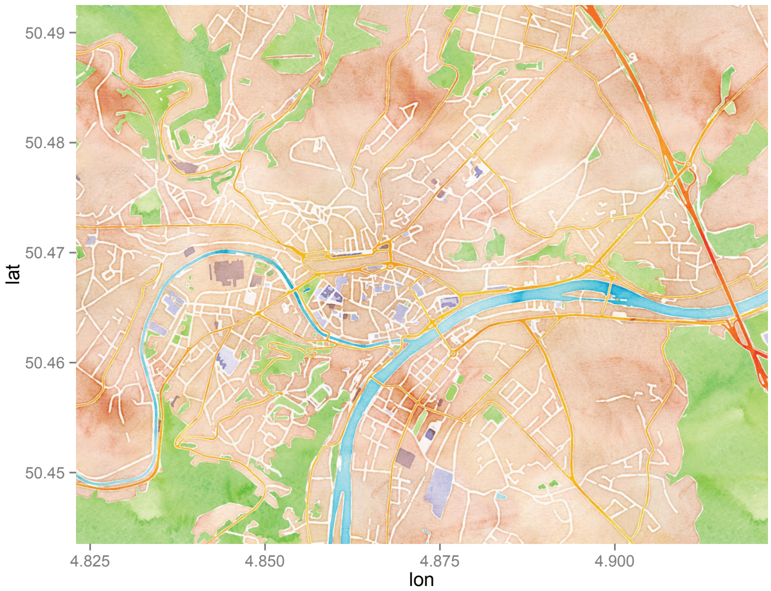
4.1 ggmap

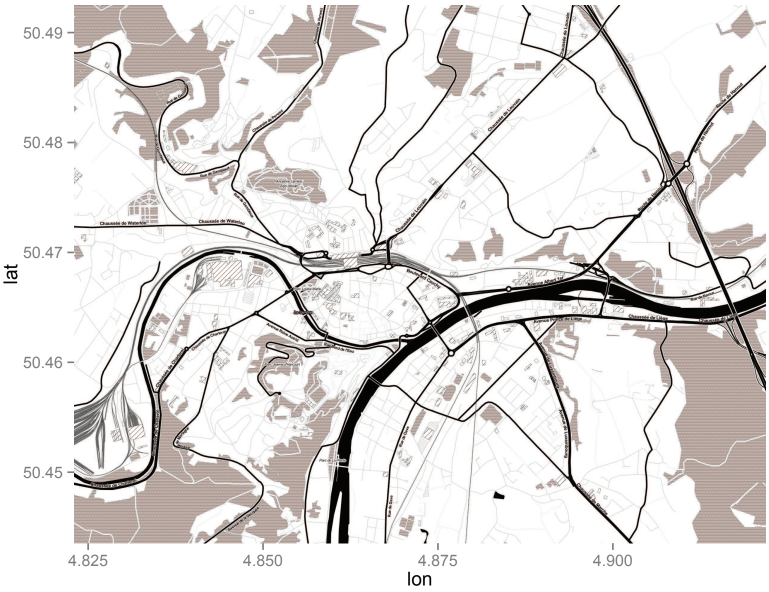

25
0
−25
−50
lat
0
30
60
90
lon
Google
Imagery ©2016 NASA, TerraMetrics


50.570
50.565
50.560
lat
4.72
4.73
lon
Google
Imagery ©2016 , Aerodata International Surveys, Cnes/Spot Image, DigitalGlobe


50.58
50.57
50.56
50.55
lat
4.70
4.71
4.72
4.73
4.74
4.75
lon
Sauvenière
Lonzée
Chapelle-Dieu
Chemin de Grand Leez
Rue du Bordia
Rue de la Pièce du Chien
Rue de Lonzée
Rue du Pont des Pages
Chaussée de Namur
Chaussée de Tirlemont
Rue Baty de Fleurus
N4


50.50
50.45
50.40
lat
4.80
4.85
4.90
4.95
lon
Saint-Denis
Warisoulx
La Bruyere
Villers-lez-Heest
Bovesse
Beauloye
Emines
Rhisnes
Rue de Rhisnes
Ville en
Wartet
Suarlée
Namur
Citadelle de Namur
Jambes
Meuse
La Plante
Floriffoux
Malonne
Pairelle
Rue de Limoy
Bûzet
Piroy
N4
N912
N934
N904
N93
N958
N80
N90
N91
N992
N954
N947
N951
N92
N941
E411
E42
A4
Google
Map data ©2016 Google


50.49
50.48
50.47
50.46
50.45
lat
4.825
4.850
4.875
4.900
lon



Packages

```
## library(maps)
## library(mapdata)
library(ggmap)
```

Functions

```
## getting coordinates of a location
COORD <- geocode('YOUR.LOCATION') # x, y coordinates

## building the map
MAP <- get_map(location=...,source=...,maptype=...)
# all the map

# location can be a string or a bounding box
# source = c("google", "osm", "stamen", "cloudmade")
# maptype = c("terrain", "terrain-background", "satellite","roadmap
    ", ...)
```

```r

# plotting the map
ggmap(MAP)
```

4.2 gpx

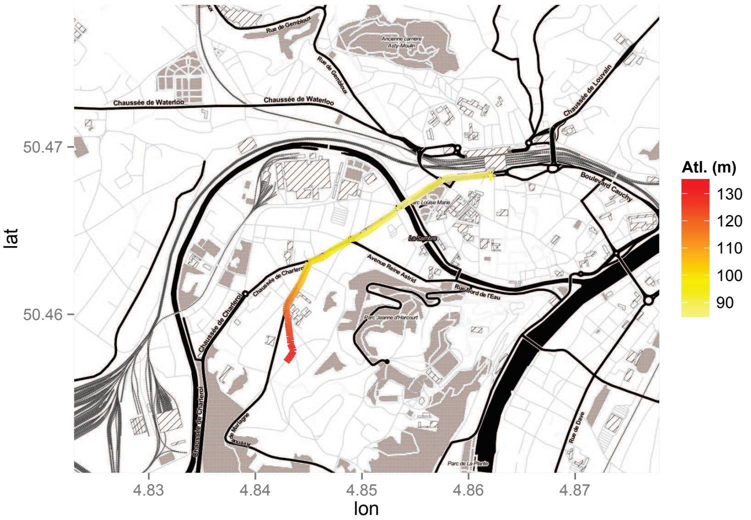
Packages

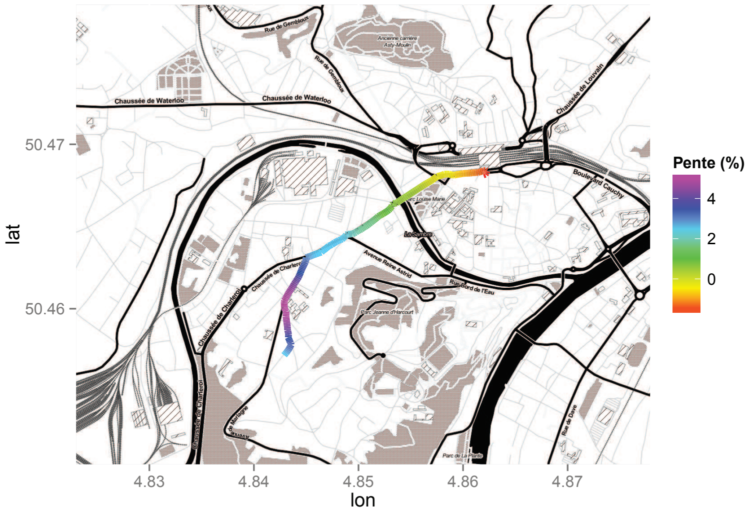
```r
library(XML)
```

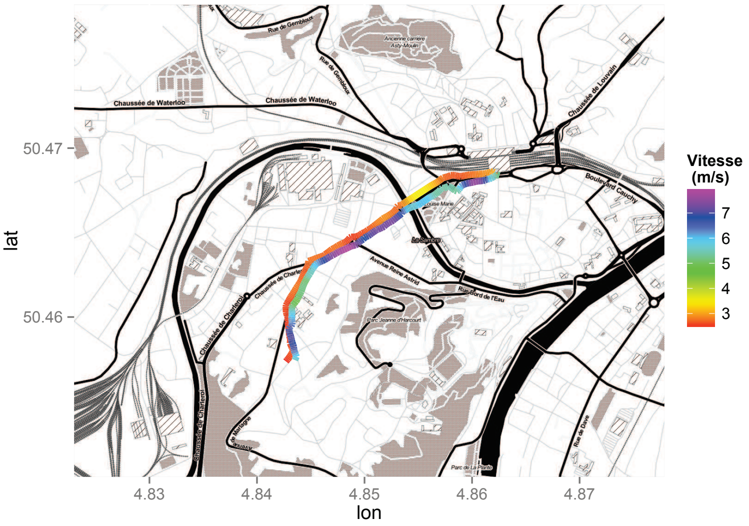
source : http://www.r-bloggers.com/stay-on-track-plotting-gps-tracks-with-r/

Procedure .gpx → data.frame (black box)

```r
gpx.gare <- htmlTreeParse("./FOLDER/FILE.gpx",error = function(...) {}, useInternalNodes = T)

## Get all elevations, times and coordinates via the respective xpath
elevations <- as.numeric(xpathSApply(gpx.gare, path = "//trkpt/ele", xmlValue))
times <- xpathSApply(gpx.gare, path = "//trkpt/time", xmlValue)
coords <- xpathSApply(gpx.gare, path = "//trkpt", xmlAttrs)

## Extract latitude and longitude from the coordinates
lats <- as.numeric(coords["lat",])
lons <- as.numeric(coords["lon",])

## Put everything in a dataframe and get rid of old variables
df.gare.retour <- data.frame(lat = lats, long = lons, ele = elevations, time = times)
```


Rue de Gembloux
Ancienne carrière Asty-Moulin
Rue de Gembloux
Chaussée de Waterloo
Chaussée de Waterloo
Chaussée de Louvain
Boulevard Cauchy
Parc Louise Marie
La Sambre
Avenue Reine Astrid
Rue Bord de l'Eau
Chaussée de Charleroi
Chaussée de Charleroi
Parc Jeanne d'Harcourt
Chaussée de Charleroi
Avenue de Marlagne
Rue de Dave
Parc de La Plante
50.47
50.46
lat
4.83
4.84
4.85
4.86
4.87
lon
Atl. (m)
130
120
110
100
90


Rue de Gembloux
Ancienne carrière Asty-Moulin
Rue de Gembloux
Chaussée de Waterloo
Chaussée de Waterloo
Chaussée de Louvain
Boulevard Cauchy
Parc Louise Marie
La Sambre
Avenue Reine Astrid
Chaussée de Charleroi
Rue Bord de l'Eau
Parc Jeanne d'Harcourt
Chaussée de Charleroi
Chaussée de Charleroi
Avenue de Marlagne
Rue de Dave
Parc de La Plante
50.47
50.46
lat
4.83
4.84
4.85
4.86
4.87
lon
Pente (%)
4
2
0



Références

Tutorial

Lovelace, R. and Cheshire, J. (2014). Introduction to visualising spatial data in R. National Centre for Research Methods Working Paper.

Unknown. Overview of Coordinate Reference Systems (CRS) in R.

Rowlingson, B., 2012. R Reference Card – Geospatial data. (`http://www.maths.lancs.ac.uk/rowlings/Teaching/UseR2012/cheatsheet.html`)

Frazier, M., –. ggmap quickstart. `https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/ggmap/ggmapCheatsheet.pdf`

Ressources web

R in general http://www.cookbook-r.com/ http://docs.ggplot2.org/current/# http://is-r.tumblr.com/

Geographical http://www.atlas-belgique.be/cms2/# http://spatialreference.org/ http://spatial.ly/

GIS and R http://r-gis.net/ http://rgeomatic.hypotheses.org/ http://www.r-bloggers.com/plot-maps-like-a-boss/ http://www.r-bloggers.com/creating-an-analysis-as-a-package-and-vignette/ http://www.r-bloggers.com/stay-on-track-plotting-gps-tracks-with-r/ http://www.56n.dk/create-your-own-hexamaps/

List of packages (check)

```
# install.packages("rgeos")
# install.packages("maps")
# install.packages("mapdata")
# install.packages("maptools")
# install.packages("mapproj")
# install.packages("raster")
# install.packages("gpclib")
# install.packages("mapproj")
# install.packages("gpclib")
# install.packages("rgdal")
# install.packages("dismo")
# install.packages("GISTools")
# install.packages("ggmap")
# install.packages("scales")
# install.packages("rworldmap")
# install.packages("rasterVis")
```

List of packages (check)

```
# install.packages("RColorBrewer")
# install.packages("gdata") ## pour read.xls
# install.packages("foreign")
# install.packages("dplyr")
# install.packages("magrittr")
# install.packages("tidyr")
# install.packages("gridExtra")
```