R & GoogleMaps

Jan Wijffels
BNOSAC: www.bnosac.be
2015

Agenda

Introduction
Overview + geonames + open geo-data
Package ggmap
Package loa & RgoogleMaps
Package plotKML
Conclusion

About me

Been using R for about 10 years, developed and co-developed some R packages related to machine learning & large data (ffbase, ETLUtils, RMyrrix, RMOA, RMETAR).
Founder of `www.bnosac.be`.

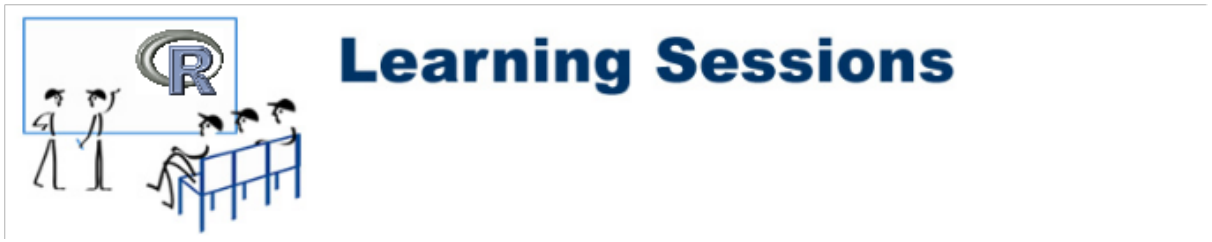
- Providing consultancy services in open source analytical engineering
- R/Oracle R Enterprise/PostgreSQL/Python/ExtJS/Hadoop/...


...

- Expertise in predictive data mining, biostatistics, geostats, python programming, GUI building, artificial intelligence, process automation, analytical web development
- R implementations & R application maintenance & R training/consulting

bnosac
open analytical helpers

R learning sessions = sharing & dissiminate knowledge



- R and GoogleMaps
- R and Robotics
- R and web development RApache, Shiny, OpenCPU
- R and text mining
- R and geostatistical modelling
- R and fMRI
- R and machine learning
- R and Bioconductor
- R and *no sales talks*, R and you ...

Main source: Journal of statistical software & R journal

Overview

When: you need georeferenced data (latitude / longitude).

- Mostly used are coordinates in the World Geodetic System (WGS) `en.wikipedia.org/wiki/World_Geodetic_System`
- Types of data
  - crime
  - distance to school
  - animals, plants
  - traffic, GPS tracks, iWatch, mobile data
  - HR data (home/work distances)
  - sports data, elections, ...
- In general: R is very good at spatial data handling (base package is the *sp* package)

Background reading

- Packages documentation of RgoogleMaps and geonames
- **ggmap**: `journal.r-project.org/archive/2013-1/kahle-wickham.pdf`
- **loa**: `www.jstatsoft.org/v63/i04`
- **plotKML**: `www.jstatsoft.org/v63/i05`
- Spatial task view: `http://cran.r-project.org/web/views/Spatial.html`

Open geographical data about Belgium - geonames

Where to get open geographical data about Belgium?

About GeoNames

The GeoNames geographical database is available for download free of charge under a creative commons attribution license. It contains over 10 million geographical names and consists of over 9 million unique features whereof 2.8 million populated places and 5.5 million alternate names. All features are categorized into one out of nine feature classes and further subcategorized into one out of 645 feature codes. (more statistics ...).
The data is accessible free of charge through a number of webservices and a daily database export. GeoNames is already serving up to over 150 million web service requests per day.

What can you do with it?

- `www.geonames.org/countries/BE/belgium.html`
- Get country, region information, postal codes
- Get nearby streets, weather, wikipedia entries, postcode, points of interest

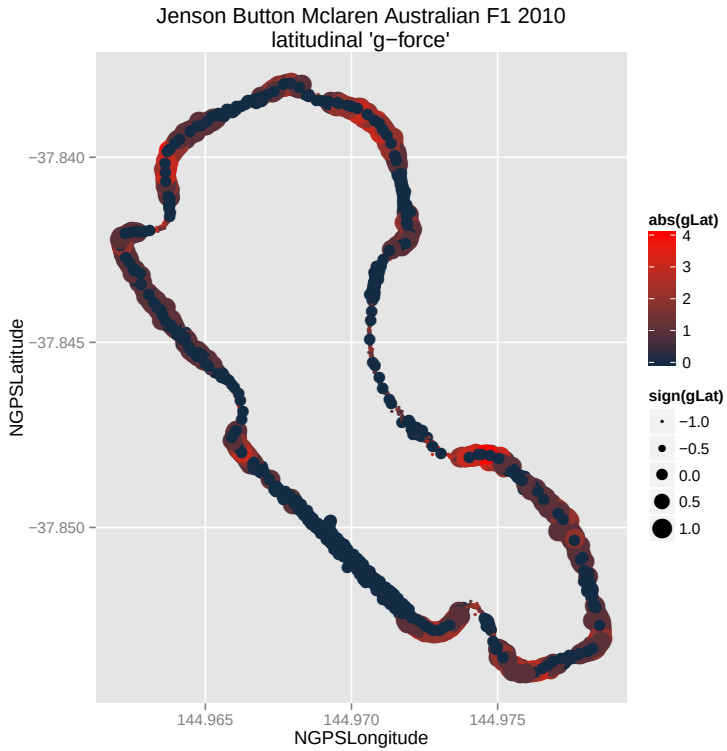
```
## Will use data available at this directory
workdir <- "C:/Users/Jan/Dropbox/Work/2015/Courses/RBelgium/RGoogleMaps"

## The geonames package
library(geonames)
options(geonamesUsername="bnosac")
GNcountryInfo(country = "BE", lang = "en")

  continent  capital         languages geonameId    south isoAlpha3     north fipsCode popu
1        EU Brussels nl-BE,fr-BE,de-BE   2802361 49.49361       BEL 51.505444       BE   10
  areaInSqKm countryCode     west countryName continentName currencyCode
1    30510.0          BE 2.546944     Belgium        Europe          EUR

##
## Give some examples based on a location
##
bnosacoffice <- list()
bnosacoffice$lon <- 4.366354
bnosacoffice$lat <- 50.86619
```

You can retrieve all kind of information about a latitude/longitude combination: **country, region, postcode, weather, wikipedia entries, streets, street intersections, points of interest**

```
result <- list()
## Gets the country, administrative zone and postcod of the lat/lon
result$country <- GNcountryCode(lat=bnosacoffice$lat, lng=bnosacoffice$lon)
result$adminzone <- GNcountrySubdivision(lat=bnosacoffice$lat, lng=bnosacoffice$lon)
result$postcode <- GNfindNearbyPostalCodes(lat=bnosacoffice$lat, lng=bnosacoffice$lon)
## Gets the weather of an automatic weather station
result$weather <- GNfindNearByWeather(lat=bnosacoffice$lat, lng=bnosacoffice$lon)
## Gets wikipedia entries which are nearby
result$wikipedia <- GNfindNearbyWikipedia(lat=bnosacoffice$lat, lng=bnosacoffice$lon,
                                          lang = 'nl')
## Gets closeby streets and street intersections
result$streets <- geonames:::gnDataFrame("findNearbyStreetsOSMJSON",
                        list(lat = bnosacoffice$lat, lng = bnosacoffice$lon,
                             radius = 1, maxRows = 20),
                        "streetSegment")
result$intersection <- geonames:::getJson("findNearestIntersectionOSMJSON",
                                          list(lat = bnosacoffice$lat,
                                               lng = bnosacoffice$lon))[["intersection"]]
## Gets closeby points of interest
result$interesting <- geonames:::gnDataFrame("findNearbyPOIsOSMJSON",
                                             list(lat = bnosacoffice$lat,
                                                  lng = bnosacoffice$lon,
                                                  radius = 1, maxRows = 50),
                                             "poi")
```

You can plot geo-data which is merely data in 2D space.

```
load(file.path(workdir, "data", "telemetryjensonbutton.RData"))
library(ggplot2)
ggplot(telemetryjensonbutton) +
  geom_point(aes(x=NGPSLongitude, y=NGPSLatitude, size=sign(gLat), col=abs(gLat))) +
  labs(title = "Jenson Button Mclaren Australian F1 2010\nlatitudinal 'g-force'") +
  scale_colour_gradient(high = "red") +
  coord_map(project="mercator")
```

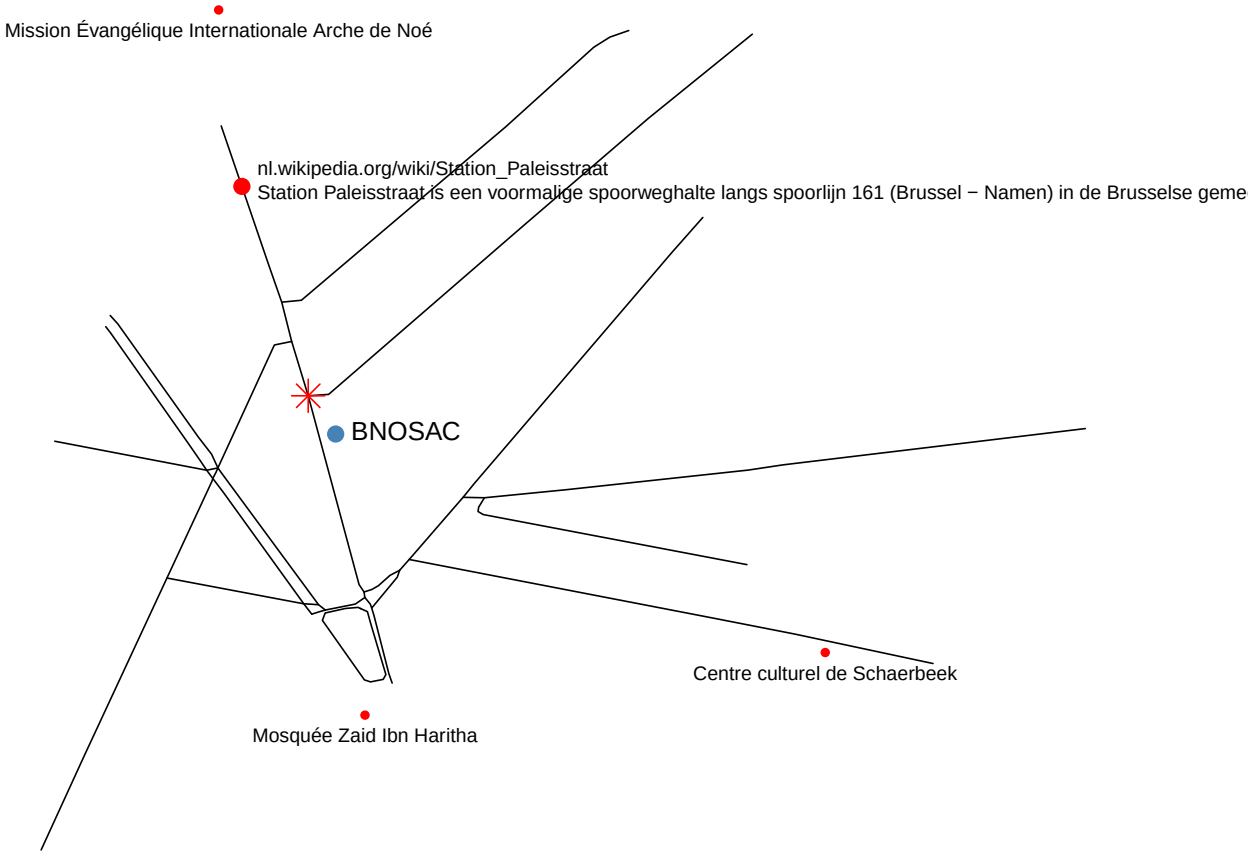



```
library(sp)
```

R has special structures to create & draw spatial lines, points, polygons, grids. Basic plotting of geo data is just **standard plotting**.

```
streetlines <- function(x){
  lapply(strsplit(x, ","), FUN=function(x){
    coords <- strsplit(x, " ")
    coords <- cbind(
      lon = as.numeric(sapply(coords, FUN=function(x) x[1])),
      lat = as.numeric(sapply(coords, FUN=function(x) x[2])))
    Lines(Line(coords), paste(x, collapse=","))
  })
}
streets <- streetlines(result$streets$line)
## Plot the streets
plot(SpatialLines(streets), main = "BNOSAC Area",
     sub = sprintf("Current temperature is %s?C",  result$weather$temperature))
## Add BNOSAC office
points(bnosacoffice$lon, bnosacoffice$lat, col = "steelblue", pch = 20, cex = 2)
text(bnosacoffice$lon, bnosacoffice$lat, "BNOSAC", pos = 4)
## Add places of worship
whatsupp <- subset(result$interesting, typeName %in% c("place_of_worship", "theatre"))
points(whatsupp$lng, whatsupp$lat, col = "red", pch = 20)
text(whatsupp$lng, whatsupp$lat, whatsupp$name, pos = 1, cex = 0.75)
## Add street intersection
points(result$intersection$lng, result$intersection$lat, col = "red", pch = 8, cex = 2)
## Wikipedia entries
points(result$wikipedia$lng[1], result$wikipedia$lat[1], col = "red", pch = 20, cex = 2)
text(result$wikipedia$lng[1], result$wikipedia$lat[1],
     sprintf("%s\n%s", result$wikipedia$wikipediaUrl[1], result$wikipedia$summary[1]),
```

BNOSAC Area



Current temperature is 12?C

Postal codes

If you are working with customer data, you probably need **post codes**. Get postal codes from Belgium available at

- http://download.geonames.org/export
- bpost.be/site/nl/residential/customerservice/search/postal_codes.html

```
x <- GNpostalCodeCountryInfo()
subset(x, countryName == "Belgium")
```

```
geodata <- list()
geodata$postcodes <- read.delim(
  file = file.path(workdir, "data", "BEpostcodes", "BE.txt"),
  header = FALSE, na.strings = "",
  col.names = c("country.code", "postal.code", "place.name",
                "admin.name1", "admin.code1",
                "admin.name2", "admin.code2",
                "admin.name3", "admin.code3",
                "latitude.wgs84", "longitude.wgs84", "accuracy"),
  encoding = "UTF-8")
head(geodata$postcodes)
```

```
  country.code postal.code                    place.name        admin.name1 admin.code
```

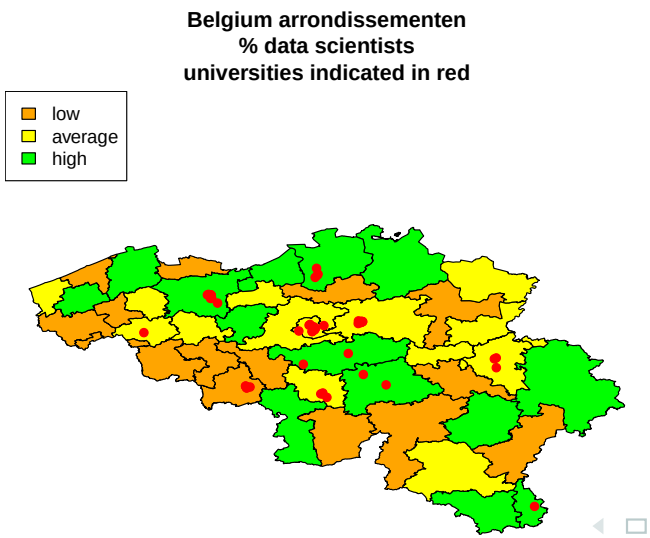
Get geographical names at download.geonames.org/export. More information at

- geonames.org/statistics/belgium.html
- geonames.org/BE/administrative-division-belgium.html

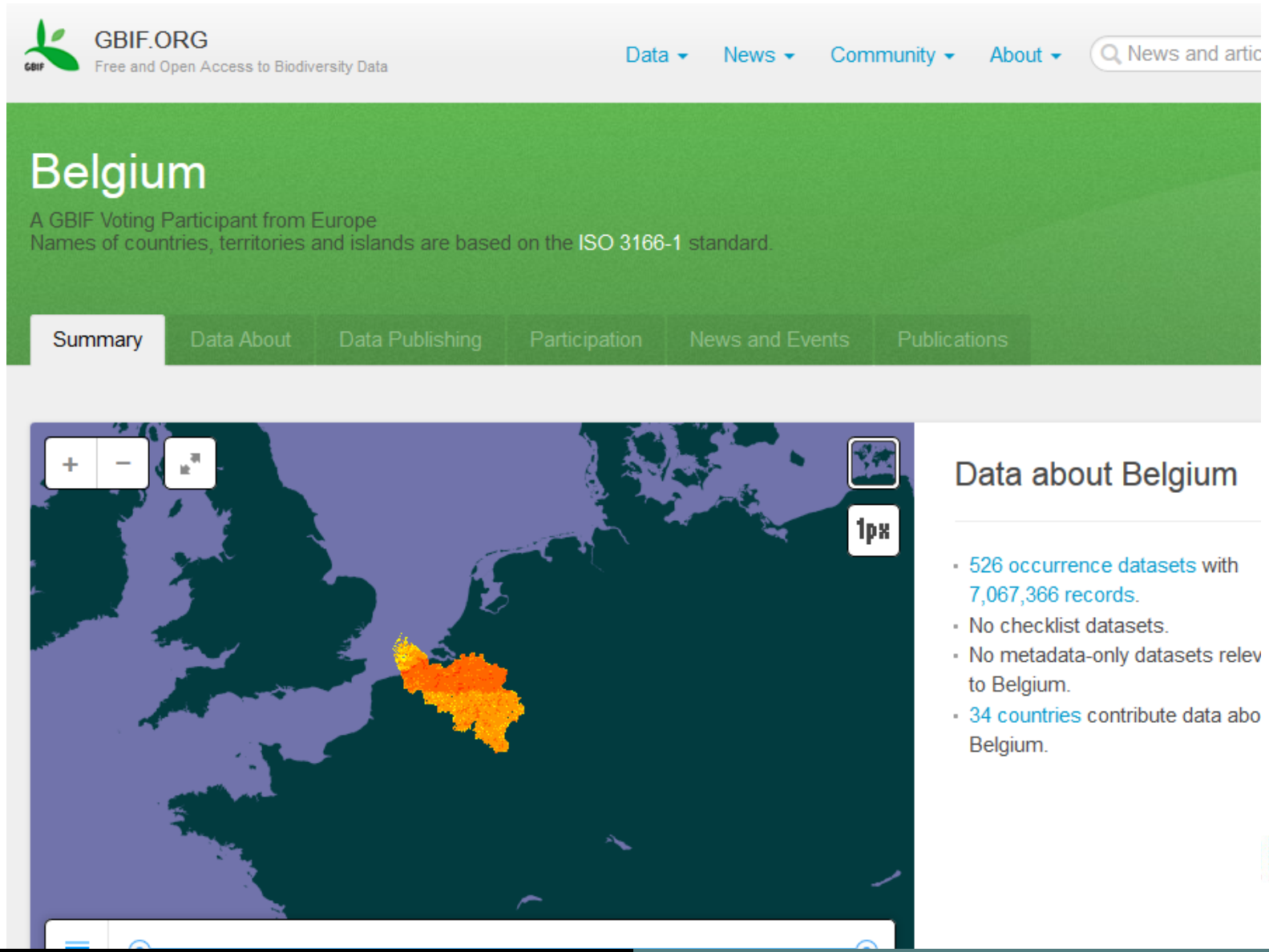
```
geodata$geo <- read.delim(
  file = file.path(workdir, "data", "BE", "BE.txt"), header = FALSE, na.strings = "",
  col.names = c("geonameid", "name", "asciiname", "alternatenames",
                "latitude", "longitude",
                "feature.class", "feature.code", "country.code", "cc2",
                "admin1.code", "admin2.code", "admin3.code", "admin4.code",
                "population", "elevation", "dem", "timezone", "modification.date"))
geodata$geofeatures <- read.delim(
  file = "http://download.geonames.org/export/dump/featureCodes_en.txt",
  header = FALSE, na.strings = "",
  col.names = c("feature", "feature.label", "feature.description"))
## Add feature definitions to the data
geodata$geo$feature <- sprintf("%s.%s",
                               geodata$geo$feature.class, geodata$geo$feature.code)
geodata$geo <- merge(geodata$geo, geodata$geofeatures,
                     by = "feature", all.x = TRUE, all.y = FALSE)
str(geodata)
```

Or you have some data from a shapefile.
www.eurogeographics.org/products-and-services/euroboundarymap

```
library(BelgiumMaps)
data(mapbelgium.arrondissementen.wgs)
plot(mapbelgium.arrondissementen.wgs,
     main = "Belgium arrondissementen\n% data scientists\nuniversities indicated in red",
     col = c("orange", "yellow", "green"))
legend("topleft",
       legend = c("low", "average", "high"), fill = c("orange", "yellow", "green"))
## Add universities
x <- subset(geodata$geo, feature.label == "university")
points(x$longitude, x$latitude, col = "red", pch = 20, cex=1.5)
```

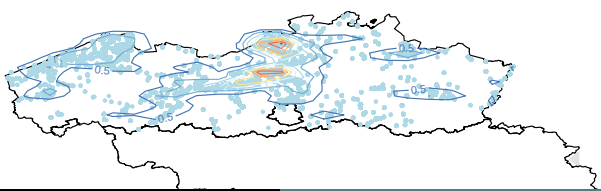


Datasets available at www.gbif.org



Waterbirds dataset: www.gbif.org/dataset/7f9eb622-c036-44c6-8be9-5793eaa1fa1e

```
library(data.table)
waterbirds <- fread(file.path(workdir, "data",
                              "dwca-watervogels-occurrences", "occurrence.txt"))
waterbirds <- subset(waterbirds, !is.na(decimalLongitude) & !is.na(decimalLatitude))
## Plot the birds
data(mapbelgium.regios.wgs)
plot(mapbelgium.regios.wgs, main = "Waterbirds density")
## sample for plotting
idx <- sample(nrow(waterbirds), 25000)
points(waterbirds$decimalLongitude[idx], waterbirds$decimalLatitude[idx],
       col = "lightblue", pch = 20, cex = 0.6)
## overlay the density contour plot
library(MASS)
dens <- kde2d(x = waterbirds$decimalLongitude[idx],
              y = waterbirds$decimalLatitude[idx], n = 25)
contour(dens, add = TRUE, col = rev(RColorBrewer::brewer.pal(10, "RdYlBu")))
```



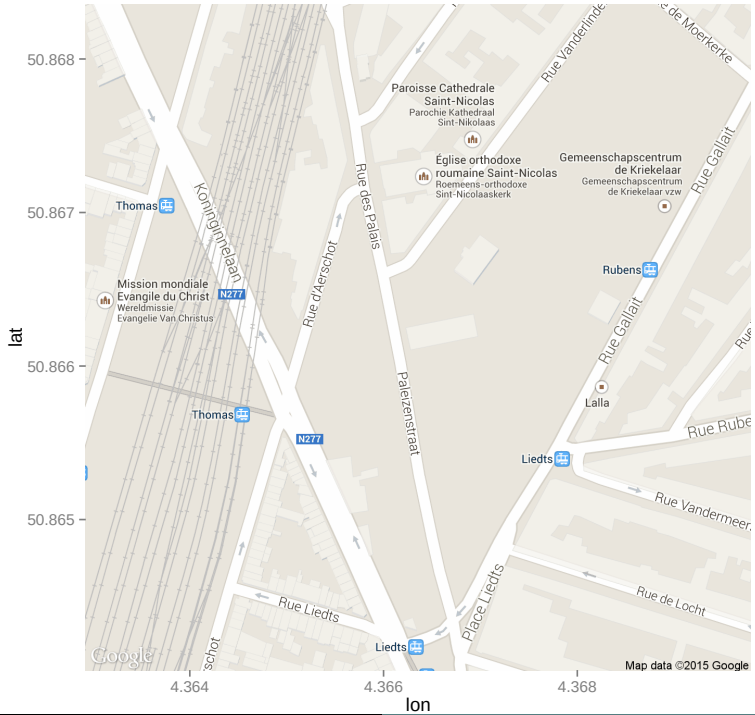
ggmap

Package ggmap:
`journal.r-project.org/archive/2013-1/kahle-wickham.pdf`
What can you do with it

- Get map with a specified center
  - google map
  - cloudemade map
  - stamen map
  - openstreet map
- geocoding & reverse geocoding
- Get a route from Google maps
- Compute map distances (driving, walking, cycling) between 2 points
- plot the map and add extra elements on to the map

Cheat sheet: `www.nceas.ucsb.edu/~frazier/RSpatialGuides/ggmap/ggmapCheatsheet.pdf`

```
library(ggmap)
x <- get_map(location = c(lon = bnosacoffice$lon, lat = bnosacoffice$lat),
             source = "google", maptype = "roadmap",
             zoom = 17, color = "color")

Map from URL :
http://maps.googleapis.com/maps/api/staticmap?center=50.86619,4.366354&zoom=17&size=%20640x
Google Maps API Terms of Service :  http://developers.google.com/maps/terms

ggmap(x)
```



Google terms of service for the free **static maps API**.
`developers.google.com/maps/documentation/staticmaps`

- 1000 Static Maps requests per IP address per 24 hour period.
- 50 Static Maps requests per IP address per minute.

Geocoding limits:
`developers.google.com/maps/documentation/geocoding`

	id	straatnaam	straatnummer	postcode
1	5450480401	Rue des Patriotes	60	1000
2	5450467064	Gutenbergsquare	22	1000
3	5450262880	Grote Haagstraat	43	1040
4	5450597524	Rue Commandant Ponthier	67	1040
5	5450470219	Kruisstraat	15	1050
6	5450342738	Verzetsplein	16	1070
7	5450391546	Paul Jansonlaan	16	1070
8	5450378699	Maurice Carêmelaan	12	1070
9	5450395192	Joseph Bracopslaan	203	1070

	id	straatnaam	straatnummer	postcode	lat	lng
1	5450480401	Rue des Patriotes	60	1000	50,8471	4,38998
2	5450467064	Gutenbergsquare	22	1000	50,8486	4,37769
3	5450262880	Grote Haagstraat	43	1040	50,836	4,39745
4	5450597524	Rue Commandant Ponthier	67	1040	50,8299	4,39987
5	5450470219	Kruisstraat	15	1050	50,8317	4,3665
6	5450342738	Verzetsplein	16	1070	50,8359	4,31125
7	5450391546	Paul Jansonlaan	16	1070	50,8348	4,30619
8	5450378699	Maurice Carêmelaan	12	1070	50,8259	4,28876
9	5450395192	Joseph Bracopslaan	203	1070	50,8283	4,28237

- 2500 requests per 24 hour period.
- 5 requests per second.

Geocoding facilities + calculating driving/walking/bicycling distances

```
## Get longitude/latitude combinations of addresses
x <- geocode(c("Paleizenstraat 153, 1030 Brussel", "V Broekaertstraat, 1090 Jette"),
             output = "more")
x[c("lon", "lat", "query")]
```

```
       lon      lat                                query
1 4.366354 50.86619 Paleizenstraat 153, 1030 Brussel
2 4.324449 50.88466    V Broekaertstraat, 1090 Jette
```

```
## Get address of longitude/latitude combination
revgeocode(location = c(bnosacoffice$lon, bnosacoffice$lat), output = "more")
```

```
                                    address street_number          route   locality adm
1 Rue des Palais 153, 1030 Schaerbeek, Belgium           153 Rue des Palais Schaerbeek
  country postal_code
1 Belgium        1030
```

```
## How far to work?
mapdist(from = "V Broekaertstraat 1, 1090 Jette", to = "Paleizenstraat 153 Brussel",
        mode = "bicycling")
```

```
                          from                          to    m    km    miles seconds
1 V Broekaertstraat 1, 1090 Jette Paleizenstraat 153 Brussel 5097 5.097 3.167276    1123
```

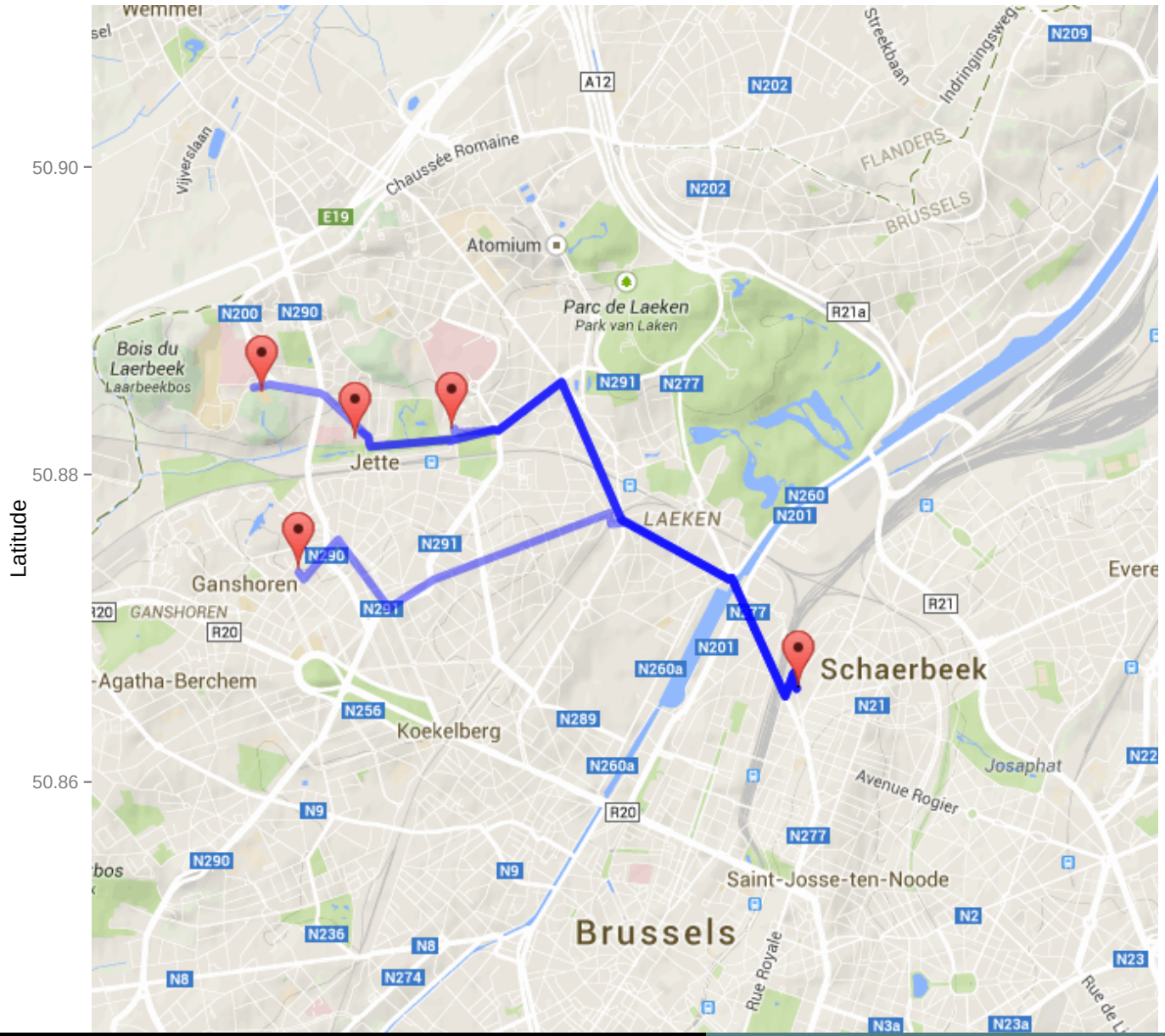
Distances (Great Circle distance) can of course also be calculated using the sp package

```
require(sp)
home <- c(bnosacoffice$lon, bnosacoffice$lat)
schools <- list()
schools$heilighart <- c(4.330608, 50.883031)
schools$poelbos <- c(4.311021, 50.885435)
schools$sintlutgardis <- c(4.314949, 50.873850)
schools$dekleinegeuzen <- c(4.320641, 50.882367)

schoolsdf <- rbind(schools$heilighart, schools$poelbos,
                   schools$sintlutgardis, schools$dekleinegeuzen)
x <- spDistsN1(schoolsdf, home, longlat = TRUE)
round(x * 1000)

[1] 3136 4444 3717 3686
```

Use route to get a route between 2 points and plot it directly

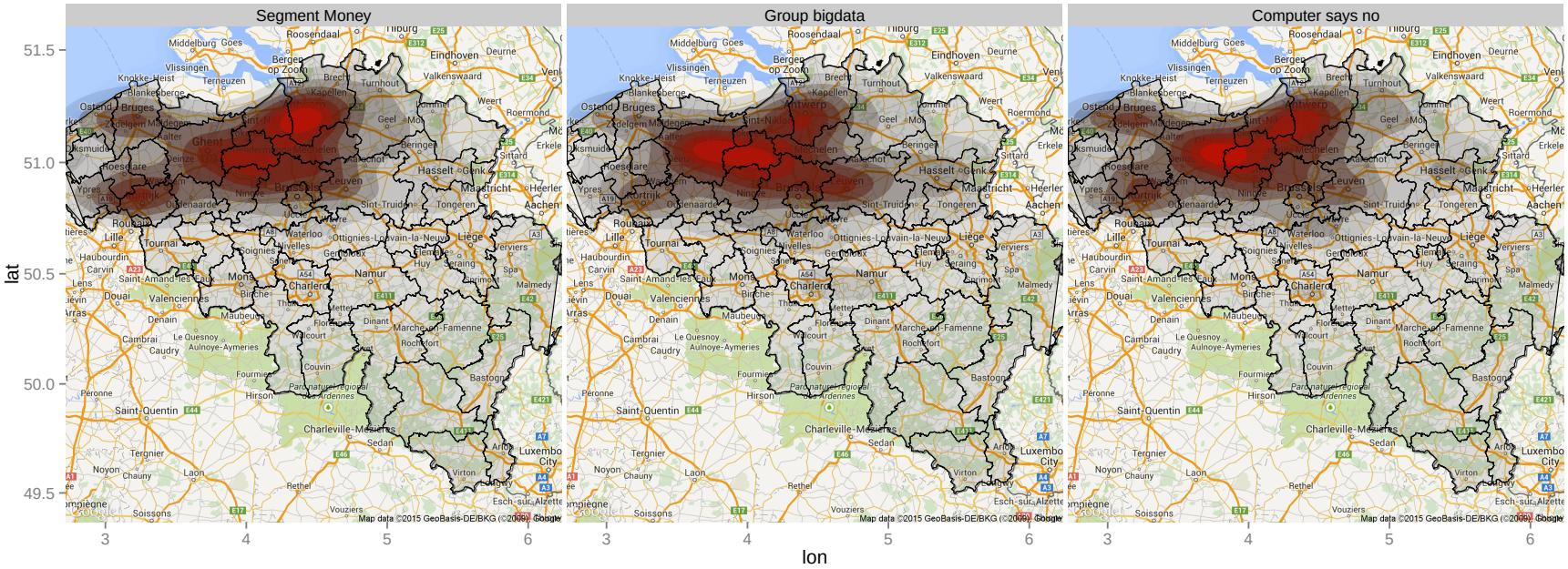
```
## Get routes to each school
schoollocations <- lapply(schools, FUN=function(x) revgeocode(x))
routes.to.school <- lapply(schoollocations, FUN=function(x, to){
  route(from = x, to = to, mode = "bicycling", structure = "route")
}, to = home)
sapply(routes.to.school, FUN=function(x) sum(x$minutes, na.rm=TRUE))

    heilighart         poelbos  sintlutgardis dekleinegeuzen
      16.30000        21.53333        19.05000        18.80000

## Plot the routes on a map
bbox <- make_bbox(lat=routes.to.school$heilighart$lat,
                  lon=routes.to.school$heilighart$lon)
mapcenter <- c(lon = mean(bbox[c("left", "right")]),
               lat = mean(bbox[c("bottom", "top")]))
x <- get_googlemap(center = mapcenter, zoom = 13, maptype = "terrain",
                   markers = as.data.frame(rbind(schoolsdf, home)),
                   path = lapply(routes.to.school, FUN=function(x) x[c("lon", "lat")]))
ggmap(x) +
  xlab("Longitude") + ylab("Latitude") +
  ggtitle("Routes to school")
```

Routes to school



ggmap are just ggplots. You can easily add extra elements to the plot

```
load(file.path(workdir, "data", "crmdata.RData"))
str(crmdata)

'data.frame': 2219 obs. of  3 variables:
 $ lat    : num  50.9 51.2 51.2 51 51.1 ...
 $ lon    : num  4.69 3.53 3.26 3.74 3.64 ...
 $ segment: Factor w/ 3 levels "Segment Money",..: 3 1 2 1 3 2 3 1 2 1 ...

## extract bounding box
belgiumbb <- GNcountryInfo(country = "BE", lang = "en")

# get a Google map with the specific bounding box
be <- get_map(loc = as.numeric(c(belgiumbb$west, belgiumbb$south,
                                 belgiumbb$east, belgiumbb$north)),
              maptype = "roadmap", source = "google")

## add a density, extra polygons from a shapefile and using facetting (group by)
shpdata <- fortify(mapbelgium.arrondissementen.wgs)
ggmap(be) +
  stat_density2d(
    aes(x = lon, y = lat, fill = ..level.., alpha = ..level..),
    size = 2, bins = 8, data = crmdata, geom = "polygon") +
  geom_polygon(aes(x = long, y = lat, group = group), data = shpdata,
               colour = 'black', fill = 'black', alpha = .1, size = .3) +
  scale_fill_gradient(low = "black", high = "red") +
  facet_wrap(~ segment)
```


Segment Money
Group bigdata
Computer says no
lat
lon
51.5
51.0
50.5
50.0
49.5
3
4
5
6
level
1.25
1.00
0.75
0.50
0.25
level
0.25
0.50
0.75
1.00
1.25
Map data ©2015 GeoBasis-DE/BKG (©2009), Google

Rgooglemaps

Package Rgooglemaps: core functionality

- Get a map using `GetMap`
- Add points on a map using `PlotOnStaticMap`
- Add polygons on a map using `PlotPolysOnStaticMap`
- easily use SpatialPoints, SpatialPolygons, SpatialLines
- package *loa* adds a function `GoogleMap`

Examples on the florabank dataset

www.gbif.org/dataset/271c444f-f8d8-4986-b748-e7367755c0c1
FLORABANK 1: a grid-based database on vascular plant distribution in the northern part of Belgium (Flanders and the Brussels Capital region)

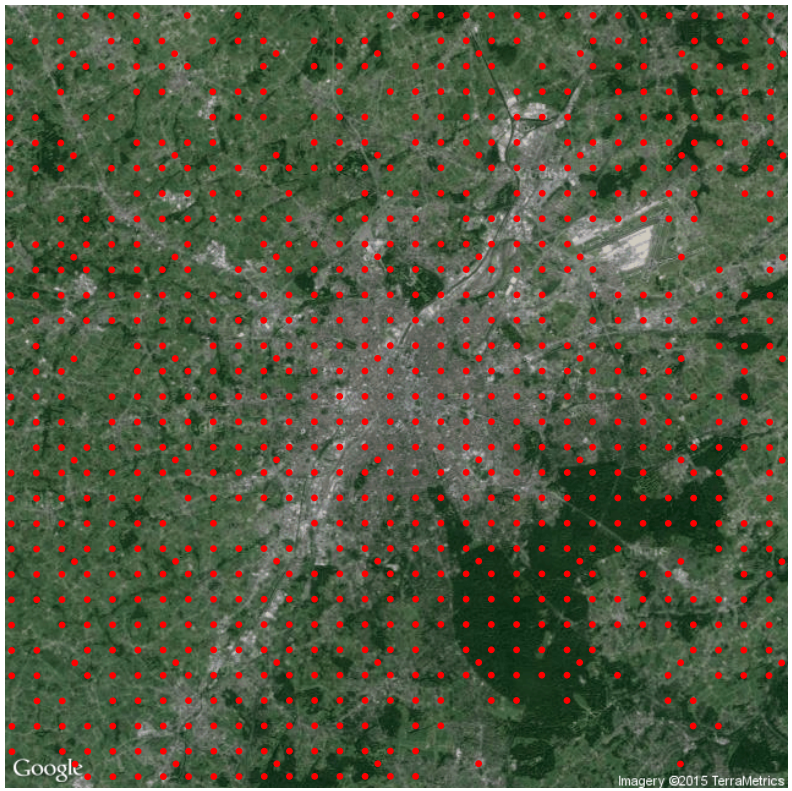
```
florabank <- fread(file.path(workdir, "data",
                             "dwca-florabank1-occurrences", "occurrence.txt"))
## Hmisc::describe(florabank)
florabank <- florabank[, c("id", "eventDate", "recordedBy", "associatedReferences",
                           "decimalLongitude", "decimalLatitude", "verbatimLocality",
                           "coordinateUncertaintyInMeters", "scientificName", "taxonRank")
                       with = FALSE]
str(florabank)

Classes 'data.table' and 'data.frame': 3365575 obs. of  10 variables:
 $ id                             : chr  "INBO:FLORA:00000001" "INBO:FLORA:00000002" "INBO:FL
 $ eventDate                      : chr  "2004-05-01/2004-05-31" "2003-01-01/2003-12-31" "200
 $ recordedBy                     : chr  "Pieter Hendrickx" "Steven De Saeger;Lieve Vriens" "
 $ associatedReferences           : chr  "Streeplijst flower 2001 (nederlandstalig)" "Biologi
 $ decimalLongitude               : num  4.15 3.79 5.41 4.61 3.75 ...
 $ decimalLatitude                : num  50.8 51 50.8 51.2 51 ...
 $ verbatimLocality               : chr  "Zuun-beekvallei-alluviale percelen" "22-lv1082" "He
 $ coordinateUncertaintyInMeters: int  700 700 700 700 700 700 700 2800 700 700 ...
 $ scientificName                 : chr  "Acer campestre" "Acer campestre" "Acer campestre" "
 $ taxonRank                      : chr  "Species" "Species" "Species" "Species" ...
```

Limit to Brussels - extracting bounding box

```
library(RgoogleMaps)
x <- get_googlemap(center = 'Brussels', zoom = 11, maptype = "satellite")
## ggmap(x)
bb <- attributes(x)$bb

## Limit data to Brussels
florabrussels <- subset(florabank,
                        decimalLongitude > bb$ll.lon & decimalLongitude < bb$ur.lon &
                          decimalLatitude > bb$ll.lat & decimalLatitude < bb$ur.lat)

## Similar to GetMap or GetMap.bbox from RGoogleMaps
x <- geocode("Brussels", output = "more")
x <- GetMap(center = c(x$lat, x$lon), zoom = 11, maptype = "satellite")
x$BBOX

$ll
       lat      lon
Y 50.71119 4.132327

$ur
       lat     lon
Y 50.98864 4.57178

x <- GetMap.bbox(lonR = range(florabrussels$decimalLongitude),
                 latR = range(florabrussels$decimalLatitude), maptype = "satellite")
PlotOnStaticMap(x)
```

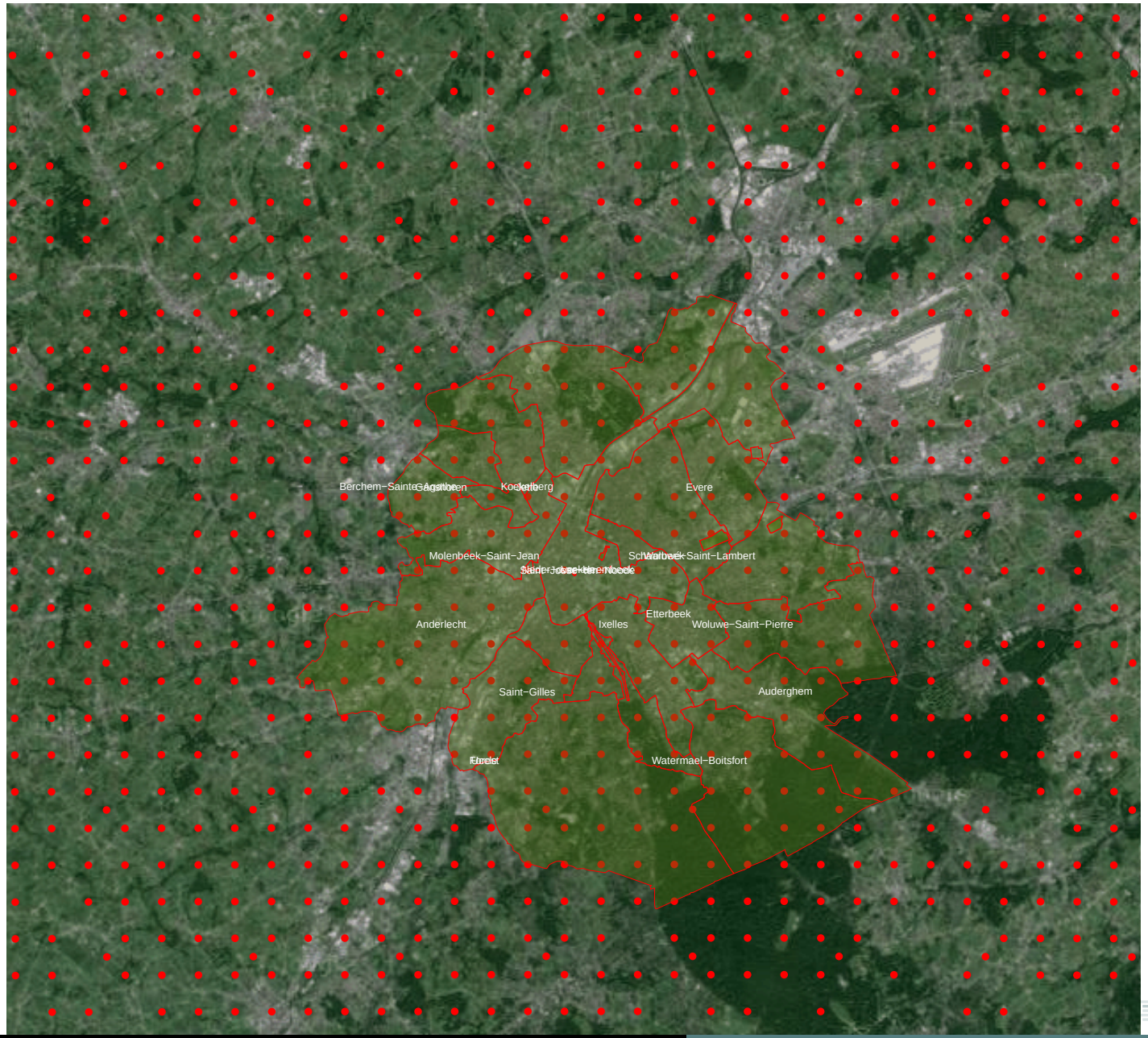
Example of PlotOnStaticMap: points plotted. For lines: uses lines, also arrows possible.

```
toplot <- florabrussels[, list(plants = .N),
                        by = list(decimalLongitude, decimalLatitude)]
toplot <- as.data.frame(toplot)
PlotOnStaticMap(MyMap = x,
                lon = toplot$decimalLongitude,
                lat = toplot$decimalLatitude,
                col = "red", pch=20, FUN=points)
```



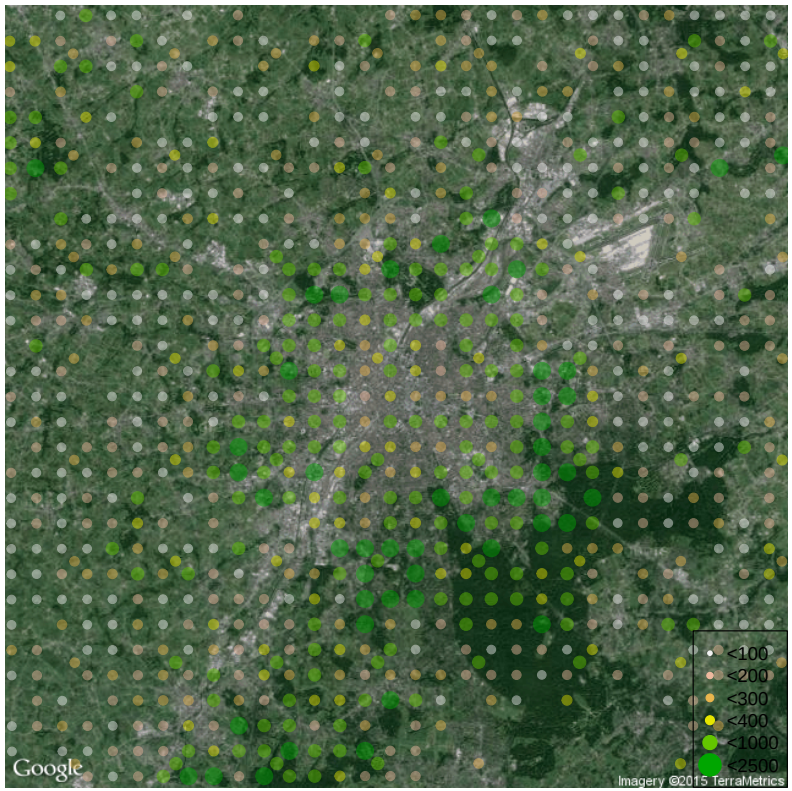
Adding polygons + text

```
library(BelgiumMaps)
library(GISTools)
data(mapbelgium.postcode.wgs)
data(provincies)
## Get polygons of postcodes in Brussels
bxlpostcodes <- subset(geodata$postcodes, admin.name1 == "Bruxelles-Capitale")
postcodespolygons <- subset(mapbelgium.postcode.wgs,
                            POSTCODE %in% bxlpostcodes$postal.code)
postcodespolygons <- SpatialPolygons(postcodespolygons@polygons,
                                     proj4string=postcodespolygons@proj4string)
## Plot grid points from the florabank
PlotOnStaticMap(MyMap = x,
                lon = toplot$decimalLongitude,
                lat = toplot$decimalLatitude, col = "red", pch=20, FUN=points)
## Plot grid postal code polygons
PlotPolysOnStaticMap(MyMap = x, polys = postcodespolygons,
                     col = add.alpha("#507415", alpha=0.4), border = "red")
## Remove postcodes which are irrelevant + add text on map
bxlpostcodes <- subset(bxlpostcodes,
                       postal.code %in% provincies$brussel$PN.Deelgemeenten &
                         !place.name %in% "Bruxelles")
TextOnStaticMap(MyMap = x,
                lat = bxlpostcodes$latitude.wgs84,
                lon = bxlpostcodes$longitude.wgs84,
                labels = gsub("Bruxelles ", "", bxlpostcodes$place.name),
                add = TRUE, cex = 0.5, col = "white")
```


Berchem−Sainte−Agathe
Koekelberg
Evere
Molenbeek−Saint−Jean
Schaerbeek
Saint−Lambert
Anderlecht
Etterbeek
Ixelles
Woluwe−Saint−Pierre
Saint−Gilles
Auderghem
Watermael−Boitsfort

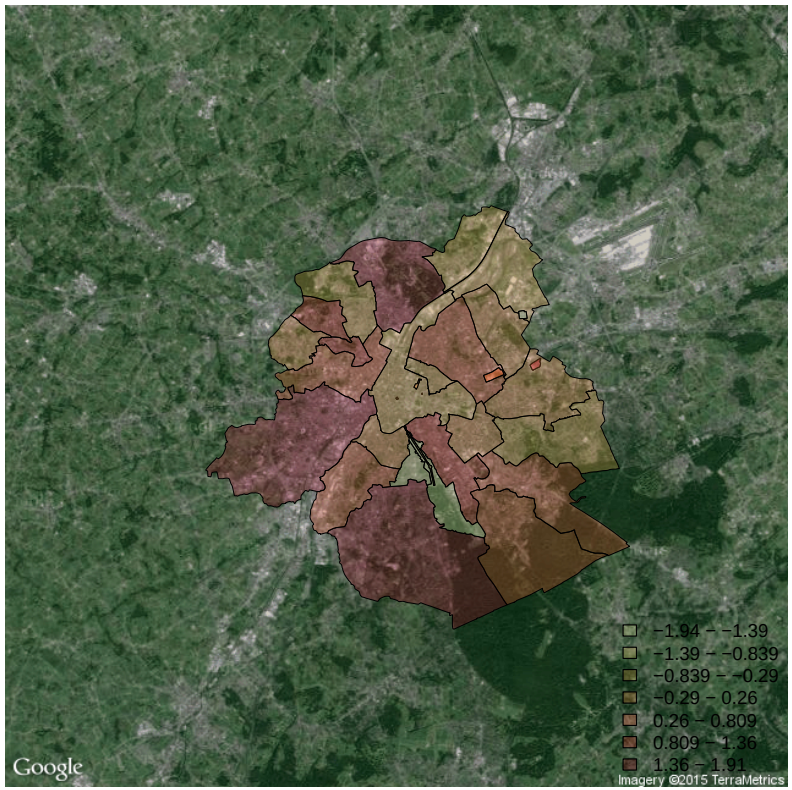
Example of bubbleMap

```
bubbleMap(SP = toplot,
          coords = c("decimalLongitude","decimalLatitude"),
          zcol = 'plants', map = x,
          LEGEND = TRUE, alpha = 0.45,
          legendLoc = "bottomright",  colPalette = rev(terrain.colors(6)),
          key.entries = c(100, 200, 300, 400, 1000, 2500))
```



Colors for each polygon

```
ColorMap(rnorm(29), map = x, polys=postcodespolygons, add = FALSE,
         alpha = 0.25, location = "bottomright")

[1] -1.9372062 -1.3879785 -0.8387508 -0.2895230  0.2597047  0.8089325  1.3581602  1.907388
```



ggmap

Package loa: `www.jstatsoft.org/v63/i04`
What can you do with it

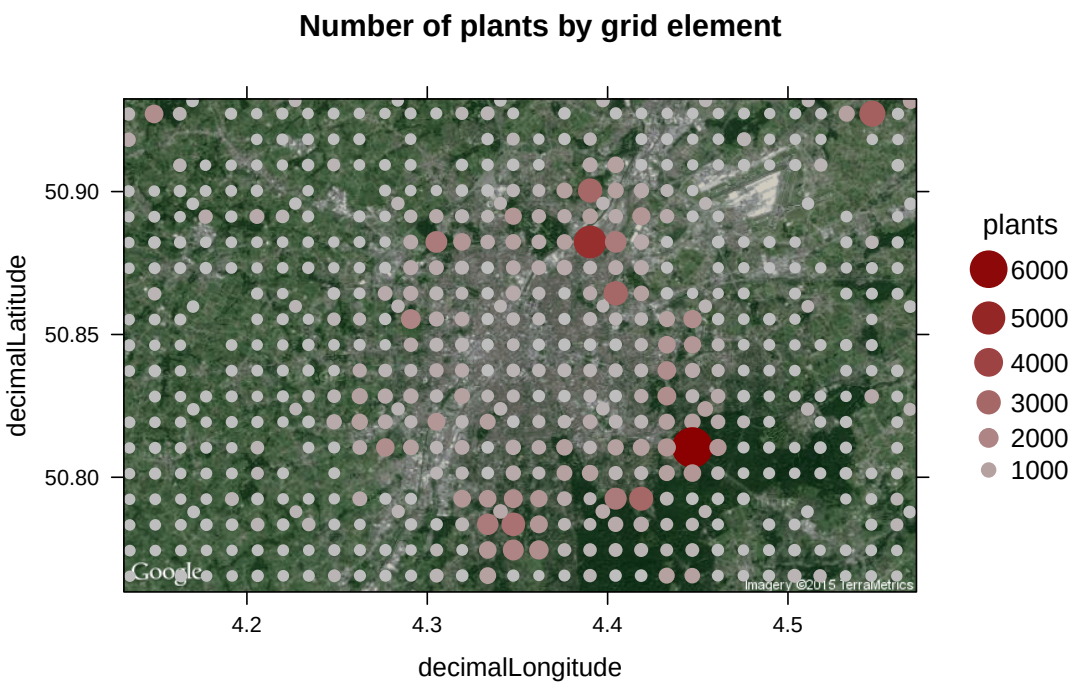
- Uses **lattice** graphics instead of ggplot
- Plotting georeferenced data on maps using lattice and RgoogleMaps
- Allow more easily to add spatial context in the plot by trellis graphics
- $z \sim lat * lon$ specification + density plots
- Enormous flexibility and **better interfacing with sp package**
- **Hot-spot analysis** & clustering of geospatial data

Examples

```
library(loa)
toplot <- florabrussels[,
                        list(plants = .N,
                             plants.netel = sum(scientificName == "Urtica dioica"),
                             plants.vlier = sum(scientificName == "Sambucus nigra"),
                             plants.es = sum(scientificName == "Fraxinus excelsior")),
                        by = list(decimalLongitude, decimalLatitude)]
toplot <- as.data.frame(toplot)

## This is a map, similar to GetMap.bbox
mymap <- makeMapArg(
  xlim = florabrussels$decimalLongitude,
  ylim = florabrussels$decimalLatitude,
  zoom = 11, lat = NULL, lon = NULL,
  latR = range(florabrussels$decimalLatitude),
  lonR = range(florabrussels$decimalLongitude),
  maptype = "satellite")
```

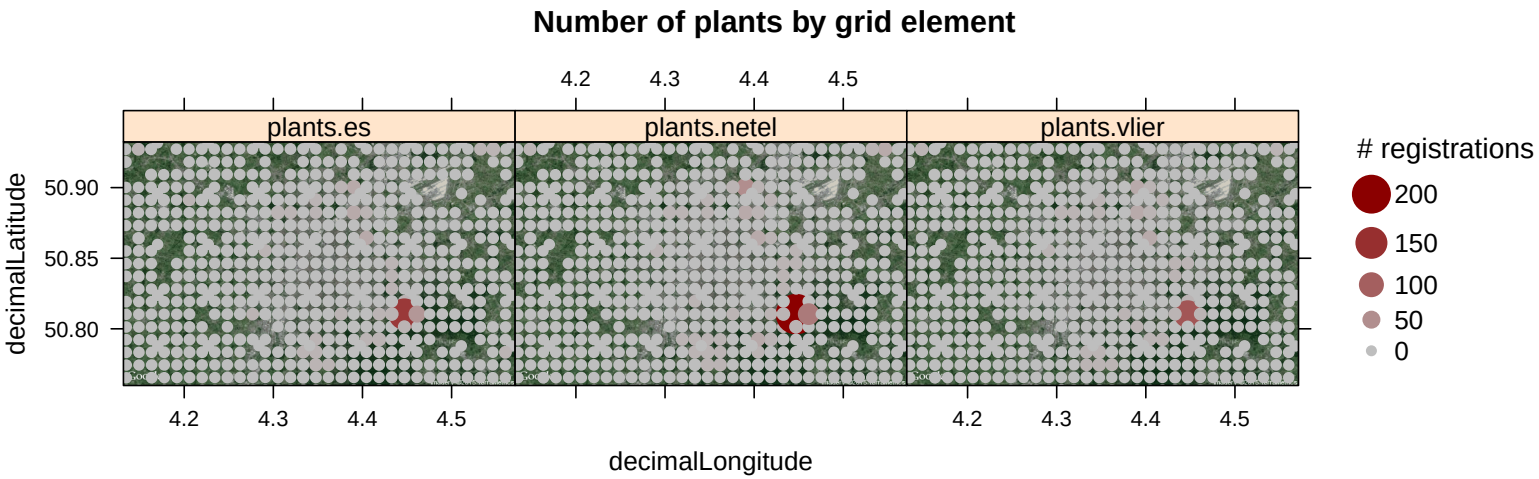
Bubble plots

```
GoogleMap(plants ~ decimalLatitude*decimalLongitude,
          data = toplot, map = mymap,
          col.regions = c("grey", "darkred"),
          main = "Number of plants by grid element")
```



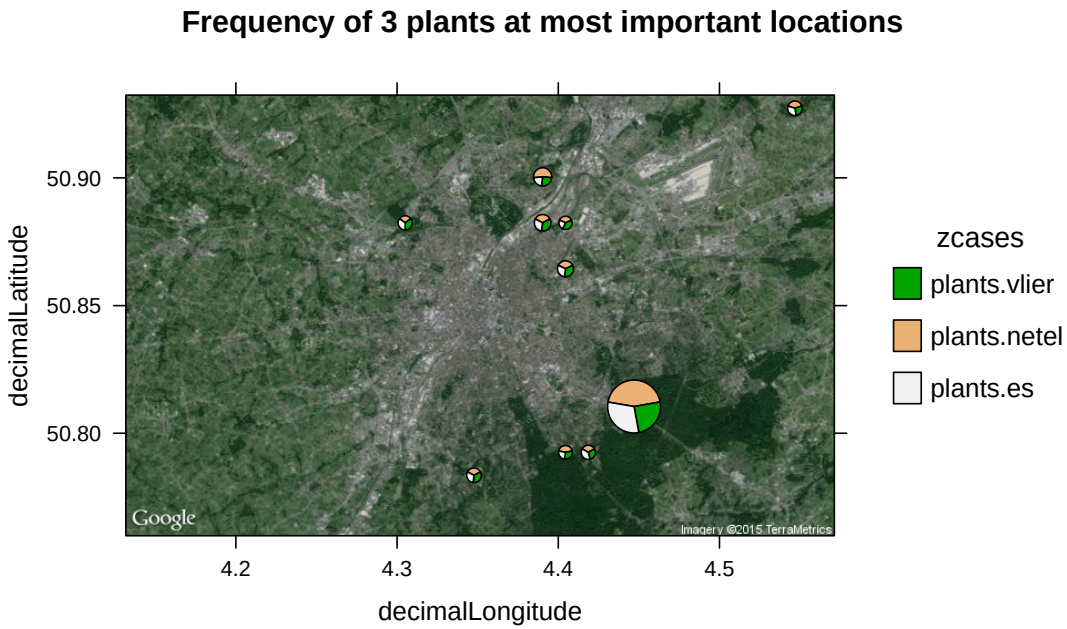
Bubble plots in a trellis

```
GoogleMap(plants.netel + plants.vlier + plants.es  ~ decimalLatitude*decimalLongitude,
          data = toplot, map = mymap,
          col.regions = c("grey", "darkred"), layout = c(3, 1),
          panel.zcases = TRUE, key.z.main="# registrations",
          main = "Number of plants by grid element")
```



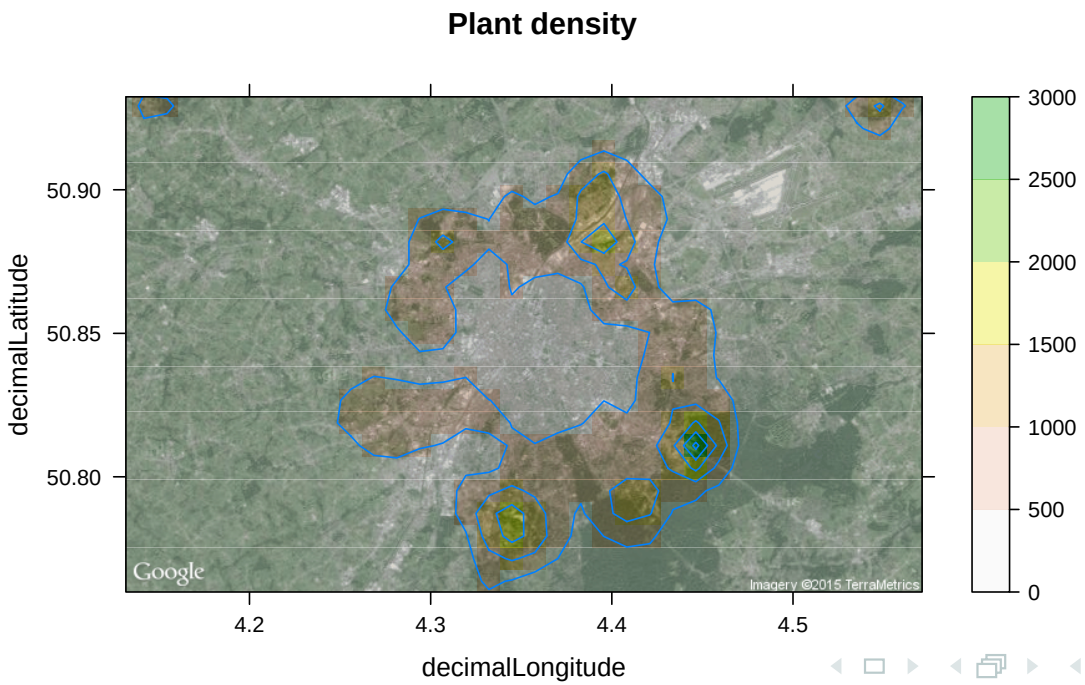
Pie charts showing distributions

```
toplot <- toplot[order(toplot$plants, decreasing=TRUE), ]
GoogleMap(plants.netel + plants.vlier + plants.es ~ decimalLatitude*decimalLongitude,
          map = mymap, panel=panel.zcasePiePlot, data=head(toplot, 10),
          col.regions = rev(terrain.colors(3)),
          main = "Frequency of 3 plants at most important locations")
```



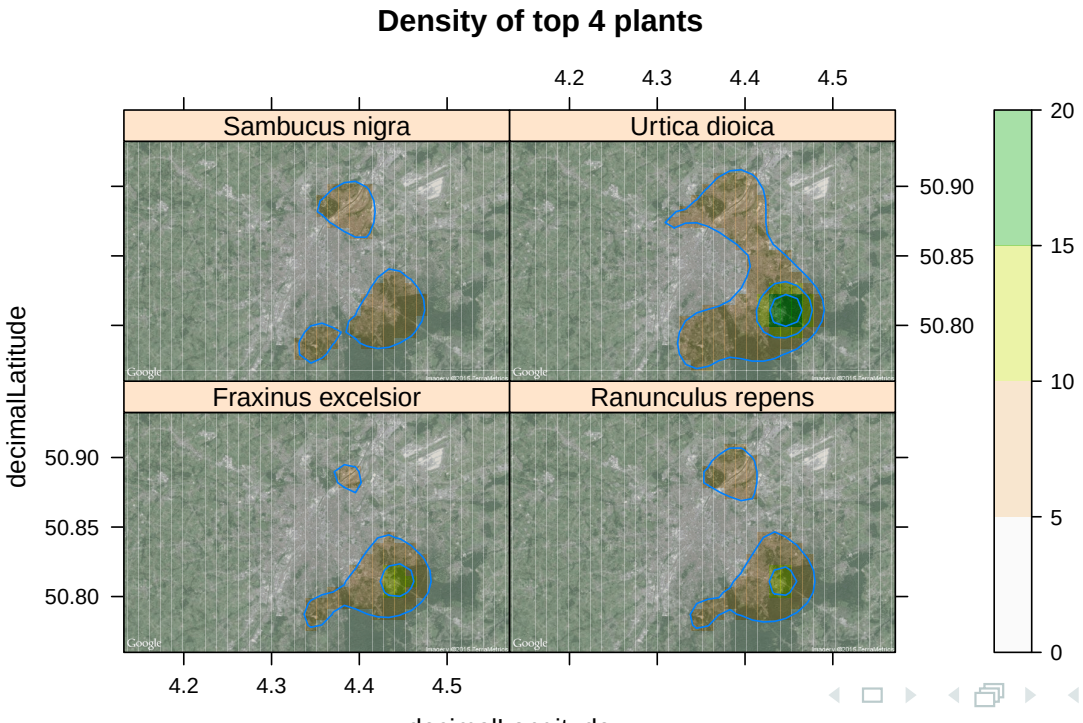
Density plots

```
florabrussels$plants <- 1
GoogleMap(plants ~ decimalLatitude*decimalLongitude,
          data = florabrussels, map = mymap,
          col.regions = rev(terrain.colors(6)), alpha.regions = 0.35,
          panel = panel.kernelDensity, n = 40,
          main = "Plant density")
```



Density plots in a trellis

```
top4plants <- head(sort(table(florabrussels$scientificName), decreasing=TRUE), 4)
GoogleMap(plants ~ decimalLatitude*decimalLongitude | scientificName,
          data = subset(florabrussels, scientificName %in% names(top4plants)), map=mymap,
          col.regions = rev(terrain.colors(6)), alpha.regions = 0.35, layout = c(2, 2),
          panel = panel.kernelDensity, n = 40,
          main = "Density of top 4 plants")
```



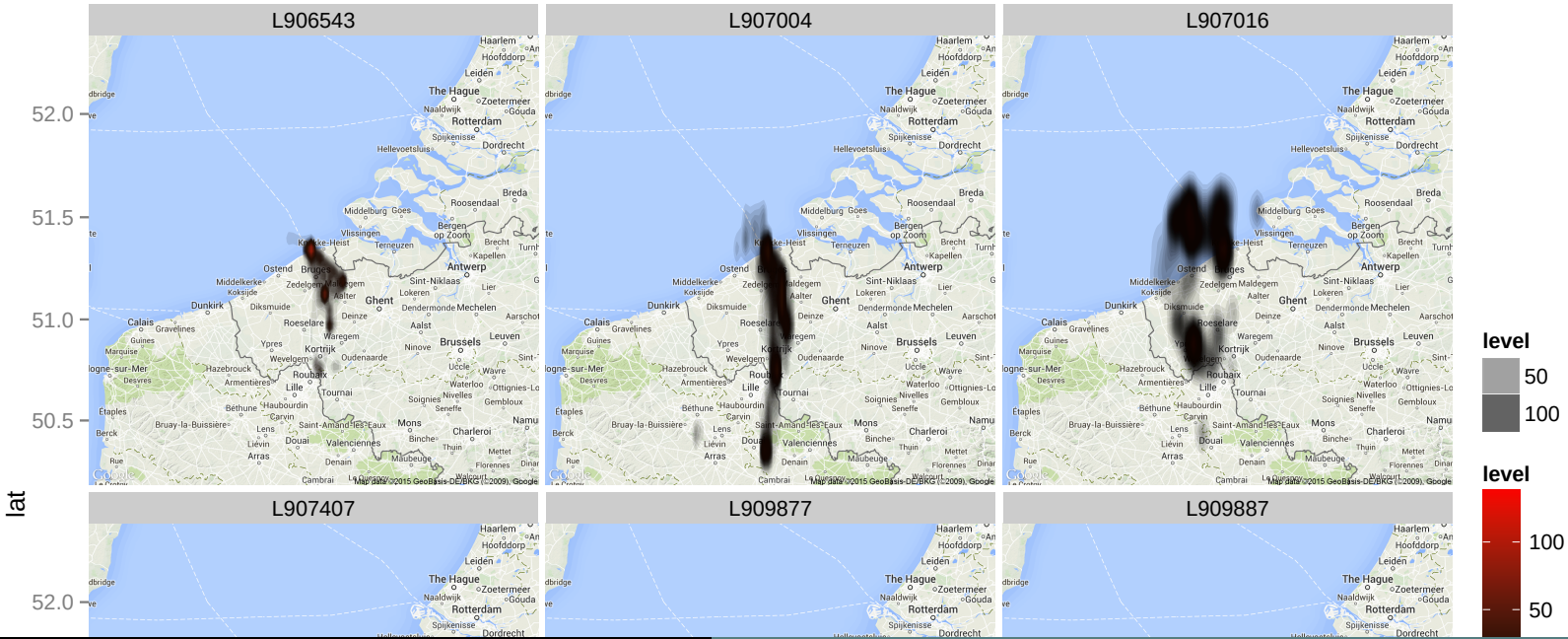
Examples on the bird tracking dataset

- www.lifewatch.be/birdmap
- lifewatch.inbo.be/blog/posts/tracking-eric.html

```
birds <- fread(file.path(workdir, "data",
                         "dwca-bird-tracking-gull-occurrences", "occurrence.txt"))
## Hmisc::describe(birds)
birds <- birds[, c("individualID", "vernacularName", "eventDate",
                   "decimalLatitude", "decimalLongitude",
                   "minimumDistanceAboveSurfaceInMeters", "minimumElevationInMeters"),
               with = FALSE]
birds$eventDate <- as.POSIXct(birds$eventDate, format = "%Y-%m-%dT%H:%M:%SZ")
birds <- as.data.frame(birds)
str(birds)
```

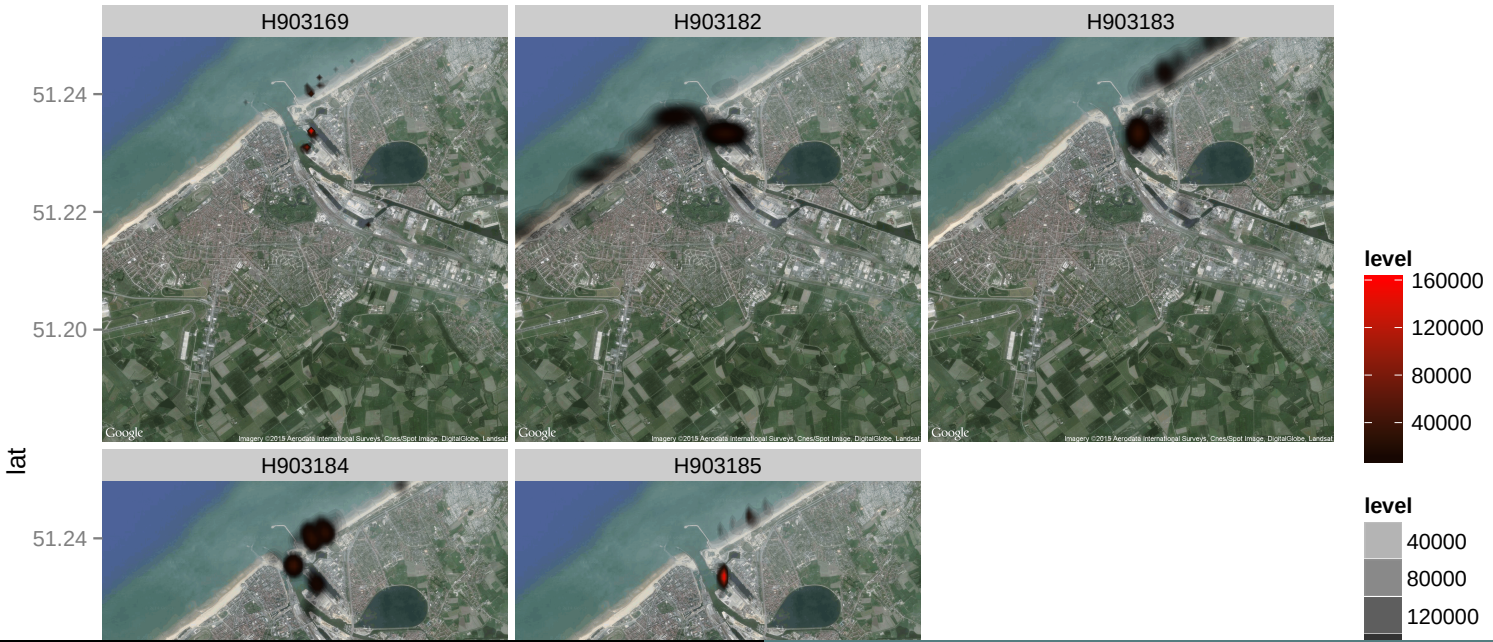
```
'data.frame': 440437 obs. of  7 variables:
 $ individualID                      : chr  "L904812" "L904812" "L904812" "L904812" ...
 $ vernacularName                    : chr  "Lesser Black-backed Gull" "Lesser Black-backe
 $ eventDate                         : POSIXct, format: "2013-05-30 18:25:07" "2013-05-30
 $ decimalLatitude                   : num  51.3 51.3 51.3 51.3 51.3 ...
 $ decimalLongitude                  : num  3.18 3.18 3.18 3.18 3.19 ...
 $ minimumDistanceAboveSurfaceInMeters: int  6 3 5 2 8 -4 -2 8 10 2 ...
 $ minimumElevationInMeters          : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
loc <- geocode("Zeebrugge Port")
mymap <- get_map(loc = c(loc$lon, loc$lat), zoom = 8,
                 maptype = "terrain", source = "google")
keep <- c('L907004', 'L907016', 'L907407', 'L909877', 'L909887', 'L906543')
toplot <- subset(birds, individualID %in% keep)
ggmap(mymap) +
  stat_density2d(
    aes(x = decimalLongitude, y = decimalLatitude,
        fill = ..level.., alpha = ..level..),
    bins = 100, data = toplot, geom = "polygon") +
  scale_fill_gradient(low = "black", high = "red") +
  facet_wrap(~ individualID, nrow = 2)
```



```
loc <- geocode("Rue de la Bassee 1, Mouscron") ## Crocky
mymap <- get_map(loc = c(loc$lon, loc$lat), zoom = 16,
                 maptype = "hybrid", source = "google")
keep <- c('L907004', 'L907407')
toplot <- subset(birds, individualID %in% keep)
ggmap(mymap) +
  stat_density2d(
    aes(x = decimalLongitude, y = decimalLatitude,
        fill = ..level.., alpha = ..level..),
    bins = 100, data = toplot, geom = "polygon") +
  scale_fill_gradient(low = "black", high = "red") +
  facet_wrap(~ individualID, nrow = 1)
```

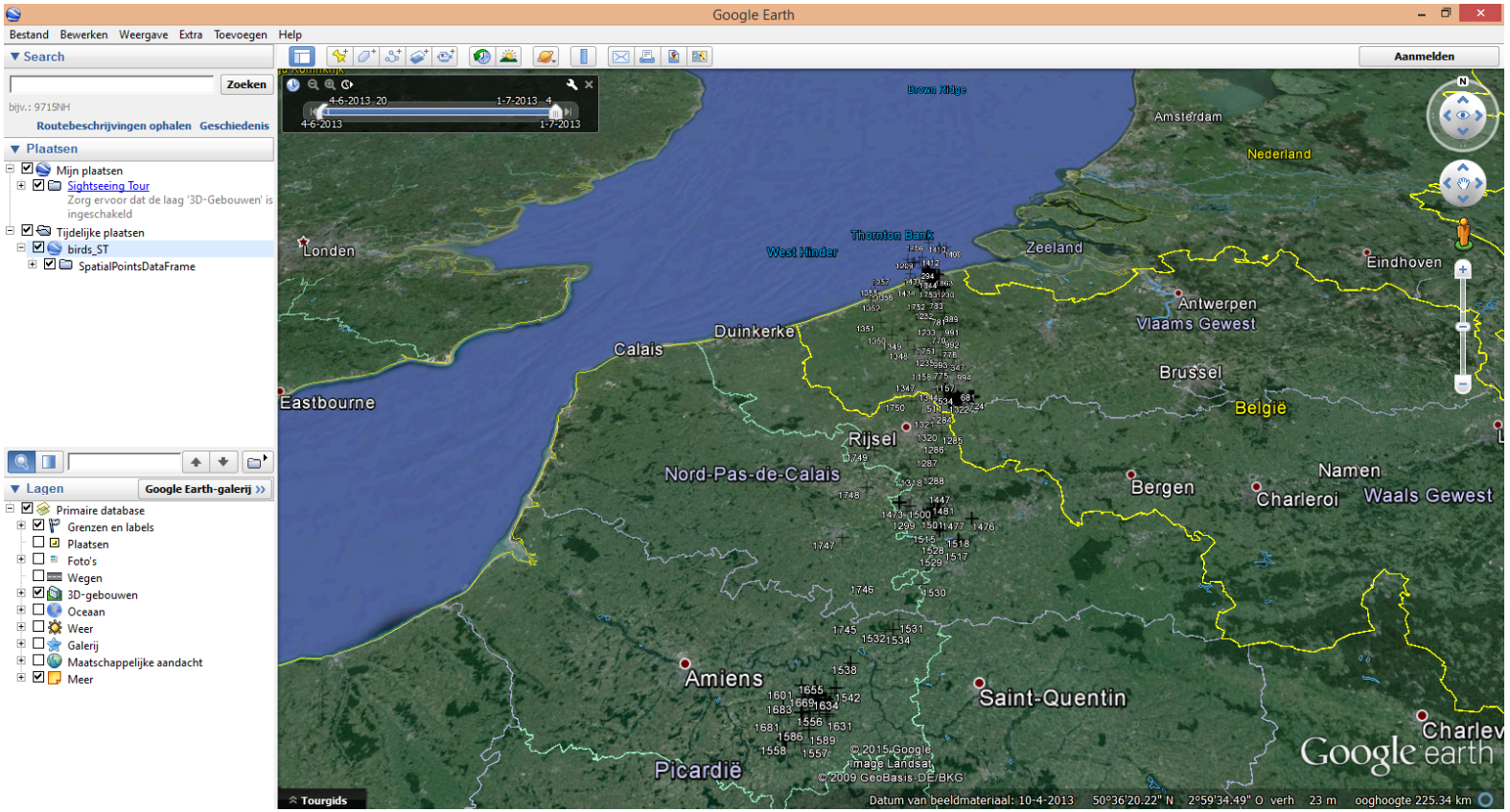
```
loc <- geocode("Ostend Harbor")
mymap <- get_map(loc = c(loc$lon, loc$lat), zoom = 13,
                 maptype = "satellite", source = "google")
toplot <- subset(birds, vernacularName %in% c("Herring Gull"))
ggmap(mymap) +
  stat_density2d(
    aes(x = decimalLongitude, y = decimalLatitude,
        fill = ..level.., alpha = ..level..),
    bins = 100, data = toplot, geom = "polygon") +
  scale_fill_gradient(low = "black", high = "red") +
  facet_wrap(~ individualID)
```



Exporting to KML - use alongside Google Earth

If you need interactivity, you can export to KML
The spacetime package allows you to indicate the timepoint of the data

```
library(sp)
library(plotKML)
library(spacetime)
birds <- SpatialPointsDataFrame(
  coords = birds[c("decimalLongitude", "decimalLatitude")],
  data = birds,
  proj4string = CRS("+proj=longlat +datum=WGS84"))
onebird <- subset(birds, individualID == "L909681")
birds_ST <- STIDF(sp=onebird, time=onebird@data$eventDate,
                  data=data.frame(ReportedDay=onebird@data$eventDate))
plotKML(birds_ST, colour_scale=SAGA_pal[['SG_COLORS_RAINBOW']])
```



Also interesting for interactive geo plots: Leaflet:
`rstudio.github.io/leaflet/`

And what does this give you?

```
require(ggmap)
x <- route(from = "Rue des Palais 153, 1030 Schaarbeek",
          to = "Haachtsesteenweg 309 1030 Schaarbeek",
    mode = "walking",
    structure = "route",
    output = c("simple", "all"), alternatives = TRUE,
    messaging = FALSE, sensor = FALSE,
    override_limit = FALSE)
x <- subset(x, route == "C")

qmap('Rue de Locht, 1030 Schaarbeek', zoom = 16, maptype = "hybrid") +
geom_path(
  aes(x = lon, y = lat), colour = 'red', size = 1,
  data = x, lineend = 'round') +
geom_point(aes(x = lon, y = lat),
           data = x[c(1, nrow(x)), ]) +
geom_text(aes(x = lon, y = lat), data = x[c(1, nrow(x)), ],
          label = c("BNOSAC", "Bar du Gaspi"), colour = "white", fontface = 2)
```

Drinks at Bar du Gaspi