Spatial Visualisations

Advances Visualisations

Jan-Philipp Kolb

Monday, October 20, 2014

Outline

Choropleth maps

Producing dotplots

Outline

The package XML

- An Introduction to the XML package for R
  http://www.omegahat.org/RSXML/Tour.pdf

XML: Tools for parsing and generating XML within R and S-Plus

This package provides many approaches for both reading and creating XML (and HTML) documents (including DTDs), both local and accessible via HTTP or FTP. It also offers access to an XPath "interpreter".

Version:	3.98-1.1
Depends:	R (≥ 1.2.0), methods, utils
Imports:	methods
Suggests:	bitops, RCurl
Published:	2013-06-20
Author:	Duncan Temple Lang (duncan@r-project.org)

Webscraping data from Wikipedia

```
library(XML)
u <- "http://de.wikipedia.org/wiki/Land_%28Deutschland%29"
tables <- readHTMLTable(u)
TabBL <- tables[[1]]
```

Webscraping data from Wikipedia

But we have the problem with the commas:

```
EW <- as.character(TabBL[,10])
EW <- gsub(",","",EW)
EW <- as.numeric(EW)
```

Example - Usage of GADM

sp is a package that provides classes and methods for spatial data.

```
library(sp)
```

Downlaod information from gadm.org:

```
con <- url("http://gadm.org/data/rda/CHE_adm1.RData")
print(load(con))
close(con)
```

A little editing

The following is necessary to combine the spatial information with the data:

```
BLn <- substr(DEU$HASC_1,4,5)
ind <- match(BLn,TabBL[,3])
ind[4] <- 4
DEU$EW <- EW[ind]
```

Excursus: the command `substr`

What we get is the follwing:

```
head(DEU$HASC_1)

[1] DE.BW DE.BY DE.BE DE.BR DE.HB DE.HH
```

So we have to use `substr`:

```
head(substr(DEU$HASC_1,4,5))
[1] "BW" "BY" "BE" "BR" "HB" "HH"
```

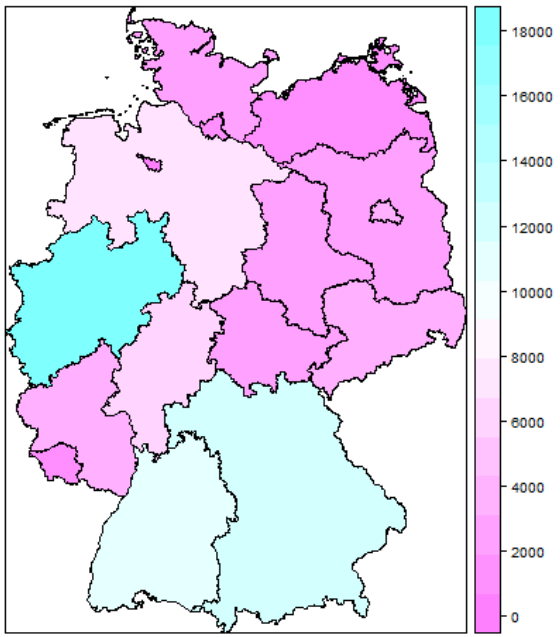
Excursus: the command `match`

```
match(c(1,2,3),c(2,1))
[1]  2  1 NA
```

```
match(c(1,2,3),c(2,1,3))
[1] 2 1 3
```

spplot population Germany

```
spplot(DEU,"EW")
```



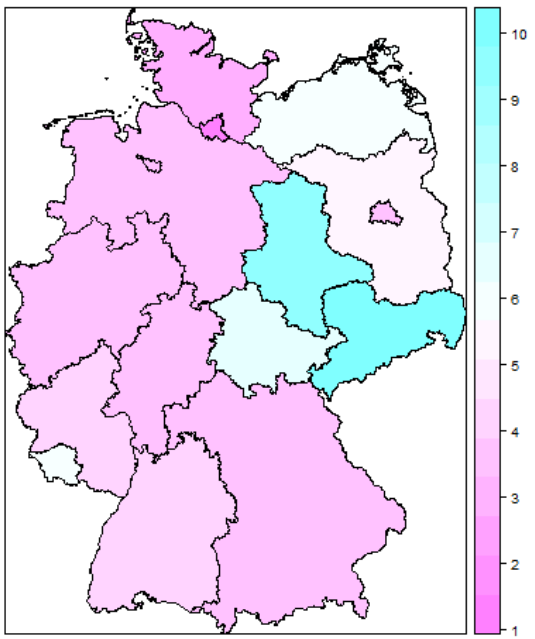
Census results - Germany

	A	B	C	D
1	Gebiet	Anteil des selbst genutzten Wohneigentums	Leerstandsquote (Wohnungen)	Anteil des zu Wohnzwecken vermieteten Wohneigentums (auch mietfrei)
2	Baden-Württemberg	51,3	4,1	44,7
3	Bayern	49,1	3,7	47,2
4	Berlin	15	3,5	81,5
5	Brandenburg	42,3	5,6	52
6	Bremen	37,8	3,6	58,6
7	Hamburg	23,7	1,5	74,8
8	Hessen	47,1	3,7	49,2
9	Mecklenburg-Vorpommern	36,2	6,2	57,7
10	Niedersachsen	52,4	3,6	44
11	Nordrhein-Westfalen	41,4	3,6	55
12	Rheinland-Pfalz	54,7	4,3	41
13	Saarland	59,4	5,7	34,9
14	Sachsen	30	9,8	60,2
15	Sachsen-Anhalt	38,3	9,4	52,3
16	Schleswig-Holstein	49,2	2,7	48,1
17	Thüringen	42,8	6,8	50,4
18				

https://ergebnisse.zensus2011.de/

Visualizing Census results

```
TabC <- read.xlsx("TabCensus.xlsx",1)
DEU$LQ <- TabC[,3]
png("spplotLQ.png")
```



A map for Europe

The easiest is to use the embedded dataset in `maptools`:

```
library(maptools)
data(wrld_simpl)
```

A map for Europe

```
country2001 <- c("Austria", "Belgium", "Switzerland",
"Czech Republic","Germany", "Denmark", "Spain",
"Finland", "France","United Kingdom","Greece",
"Hungary", "Ireland", "Israel", "Italy",
"Luxembourg","Netherlands", "Norway", "Poland",
"Portugal","Sweden", "Slovenia")
```

A map for Europe

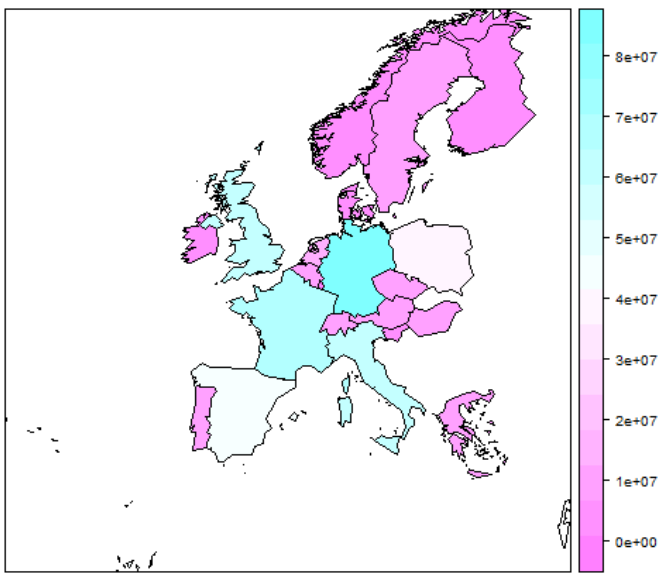
```
ind <- match(country2001,wrld_simpl$NAME)
Europe <- wrld_simpl[ind,]
plot(Europe)
```

A map for Europe - Population 2013

```
DatE <- read.xlsx("demo_pjan.xls",1)
DatE$GEO.TIME <- as.character(DatE$GEO.TIME)
DatE$GEO.TIME[11] <- "Germany"

ind2 <- match(Europe$NAME,DatE$GEO.TIME)

Europe$Pop <- as.numeric(as.character(DatE$X2013[ind2]))
```

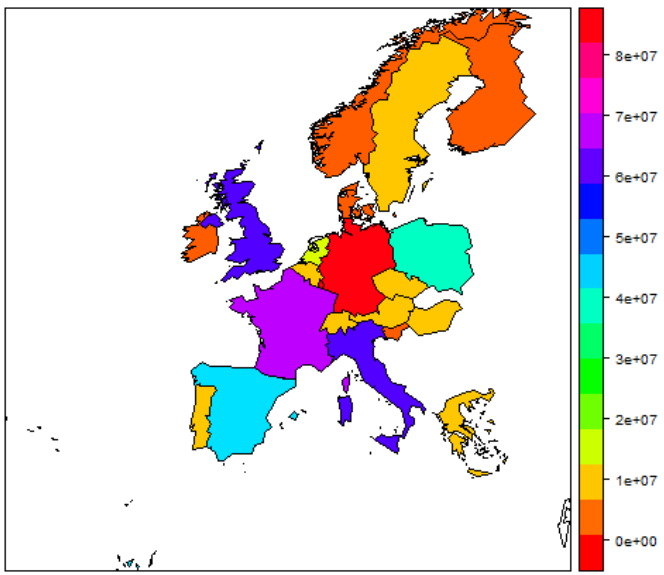
A map for Europe - Population 2013

```
spplot(Europe,"Pop")
```



A map for Europe - Population 2013

```
spplot(Europe,"Pop",col.regions=rainbow(100))
```



Outline

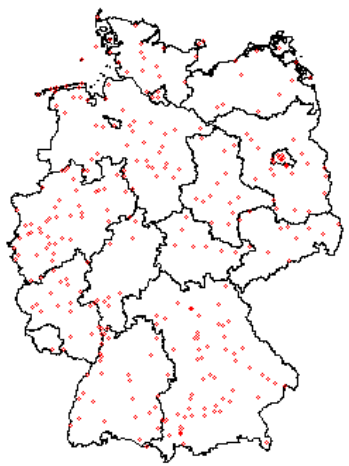
Dot map with airports

- Source code:
  `http://www.milanor.net/blog/?p=594`
- Data comes from:
  `http://openflights.org/data.html`

```
air <- read.table("airports.dat",sep=",")
colnames(air) <- c("ID", "name", "city", "country",
"IATA_FAA", "ICAO", "lat", "lon", "altitude", "timezone",
"DST","WE")

airG <- air[air$country=="Germany",]
```

Dot map with airports

```
plot(DEU)
points(airG$lon, airG$lat, col = "red", cex = .6)
```



R-package ggmap

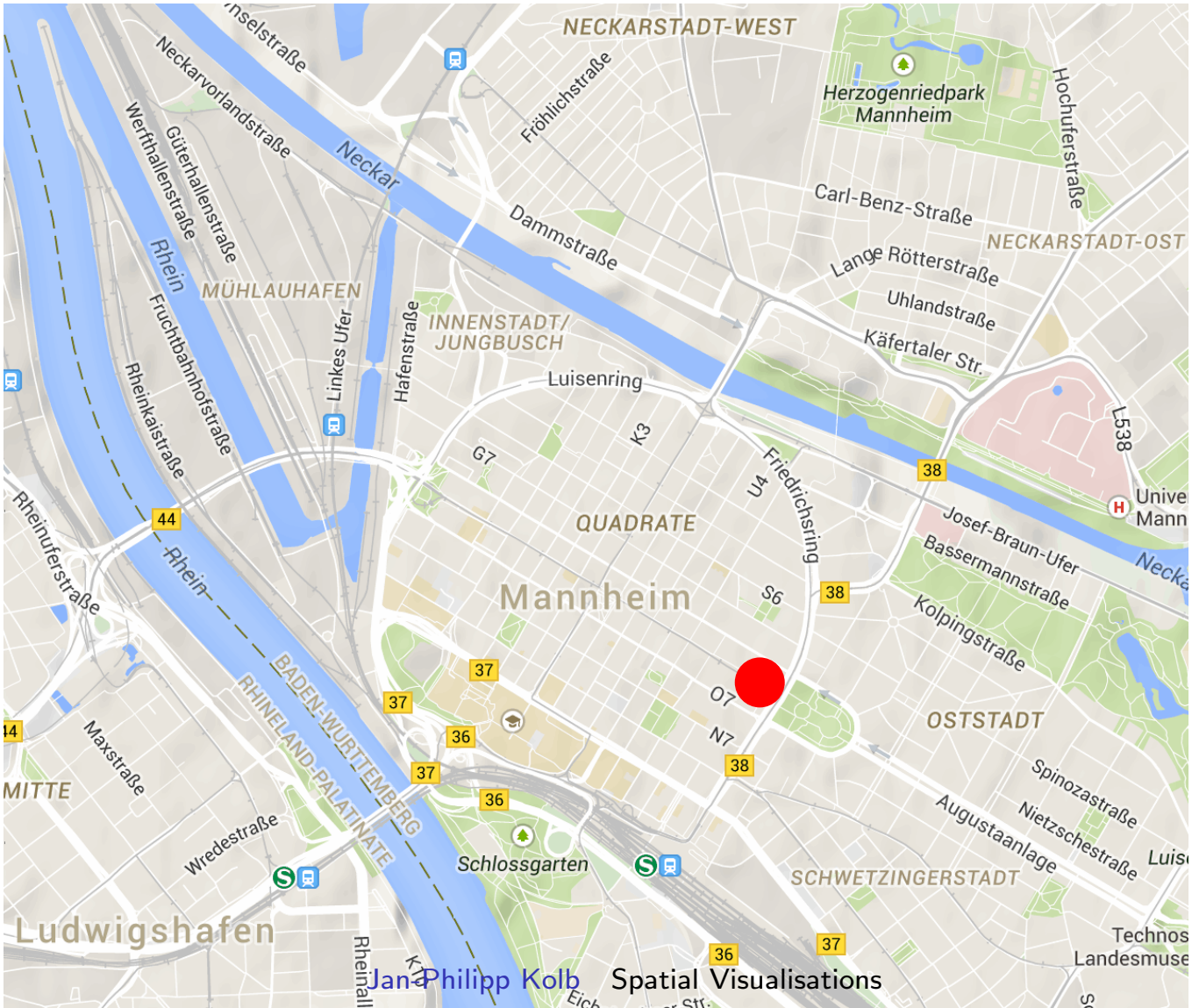
Geocoding more than one point of interest:

```
POI <- c("B2, 1 Mannheim","Hauptbahnhof Mannheim")

ListPOI <- data.frame(lat=NA,lon=NA)

for ( i in 1:length(POI)){
  geoPOI <- geocode(POI[i])
  ListPOI[i,"lat"] <-  geoPOI$lat
  ListPOI[i,"lon"] <-  geoPOI$lon
}
```

R-package ggmap

```
map <- qmap(location = 'Mannheim', zoom = 14)
address <- geocode("Mannheim Wasserturm")

map + geom_point(data = address,
            aes(x = lon, y = lat),
            color = "red",
            size = 10)
```

R-package ggmap



Exercise:

- Have a look at:

  `http://pakillo.github.io/R-GIS-tutorial`

- and tell me what would be interesting for you.