Quick high quality maps with R

Jan-Philipp Kolb

July 7 2021

Motivation

Jan-Philipp Kolb @JanPhilippKolb · 9. Nov. 2020

My Day9 #30daymapchallenge. A monochrome map of #Trier in Germany. I made this map with the #rstats **#osmplotr** package.

3 23

About the tutorial

- Four sections with each 20 min presentation and 10 minutes testing time for you

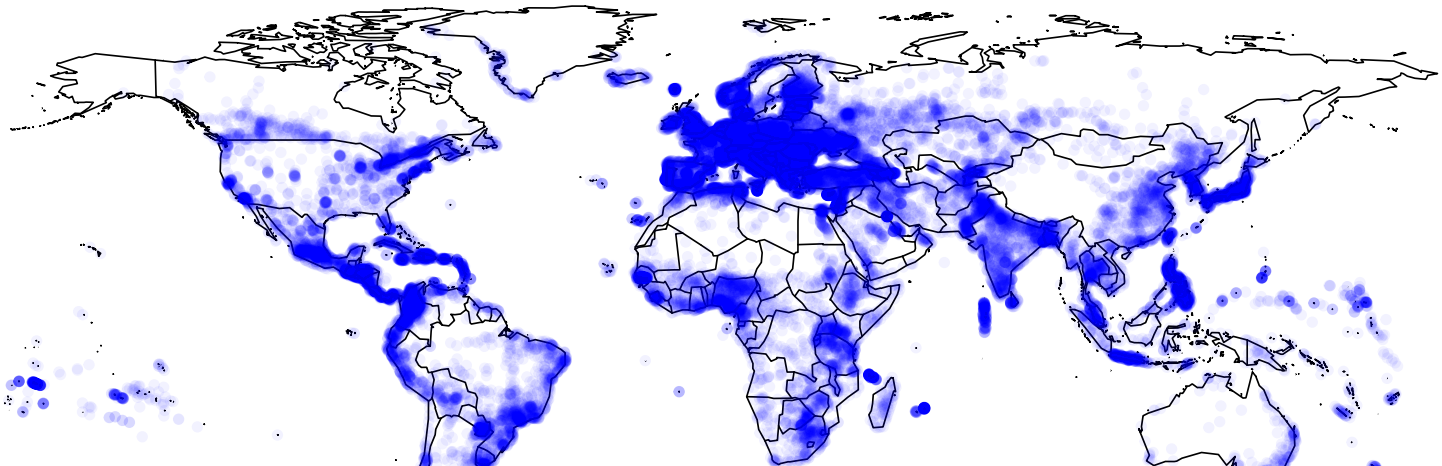
Sections

1. Focus is on quick maps - I will present some classics
2. Simple Features in R - the new way to work with geodata in R
3. OSM data in R
4.

A classic - the maps package

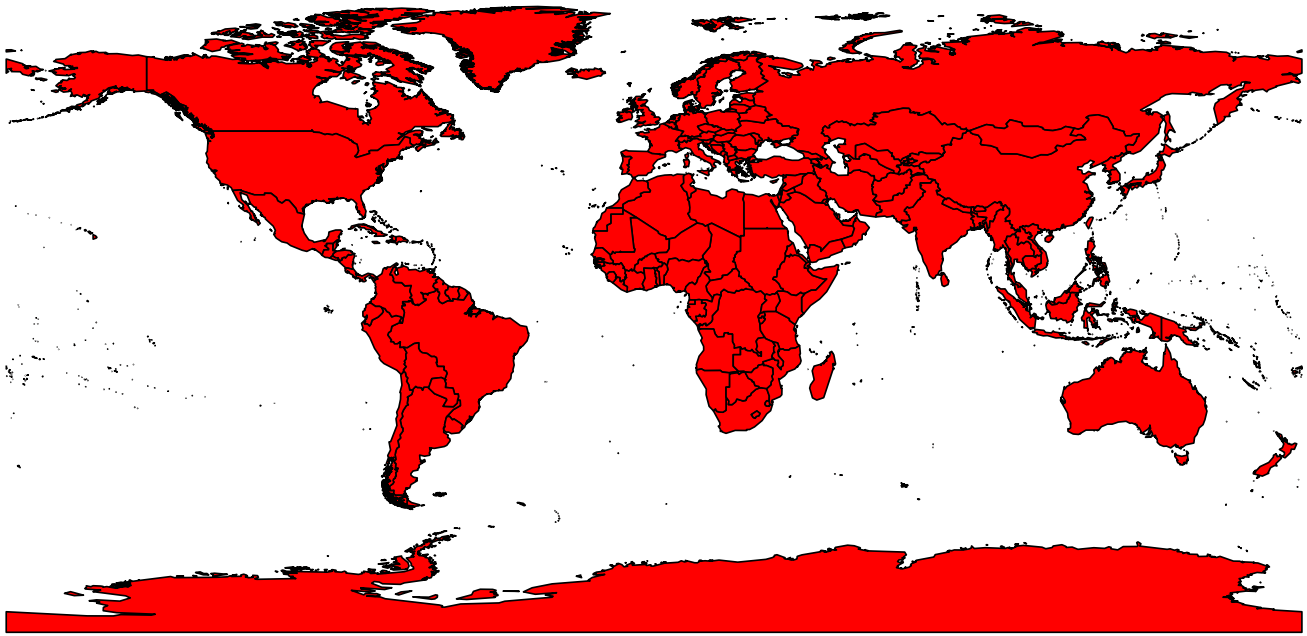
- maps package by Richard A. Becker, Allan R. Wilks and Ray Brownrigg

```
library(maps)
data(world.cities)
map()
map.cities(world.cities,col=rgb(0,0,1,.05),cex=2,pch=
```



A second classic - maptools package


```
library(maptools)
data("wrld_simpl")
plot(wrld_simpl,col="red")
```



The wrld_simpl package


```
head(wrld_simpl@data)
```

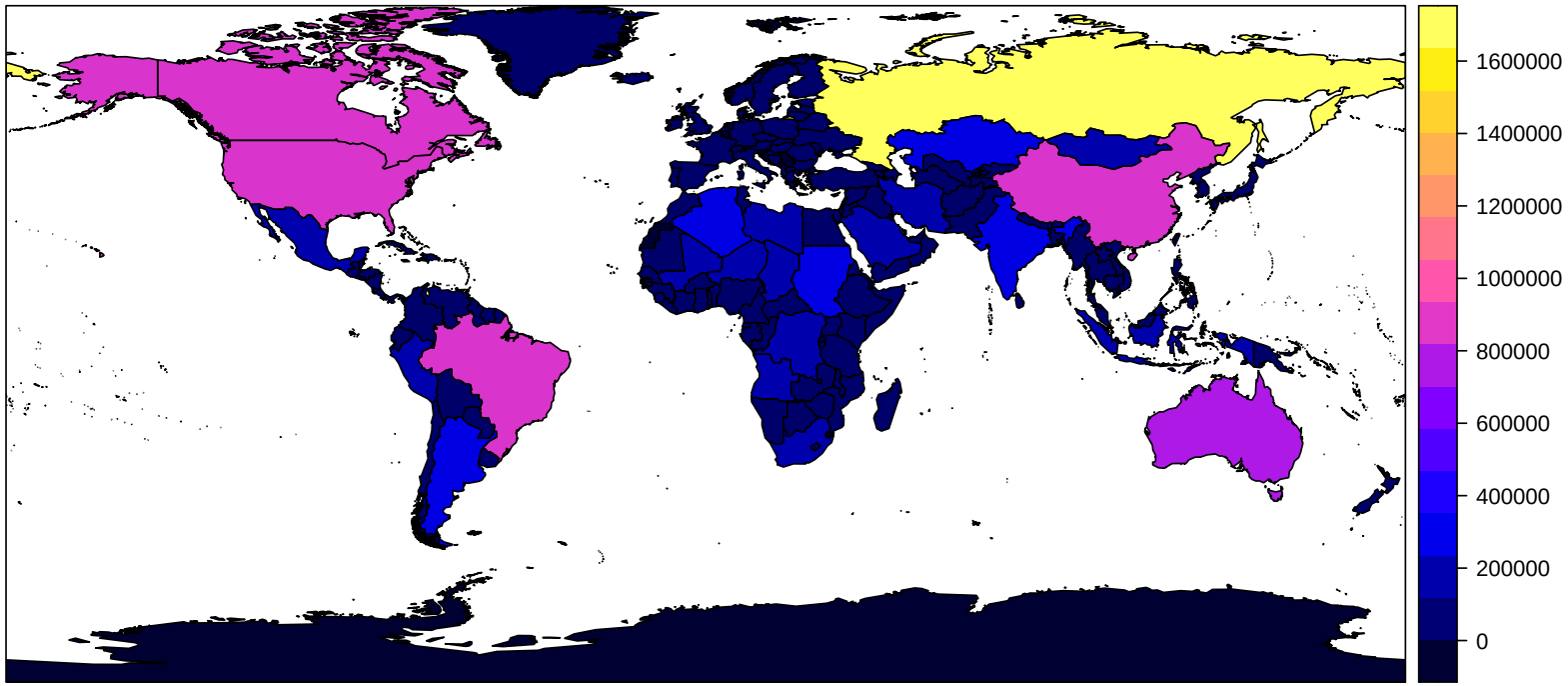
RPubs by RStudio

Show 10 entries Search:

	FIPS	ISO2	ISO3	UN	NAME	AREA	POP2005	REGION	SUBREGION	LON	LAT
ATG	AC	AG	ATG	28	Antigua and Barbuda	44	83039	19	29	-61.783	17.078
DZA	AG	DZ	DZA	12	Algeria	238174	32854159	2	15	2.632	28.163
AZE	AJ	AZ	AZE	31	Azerbaijan	8260	8352021	142	145	47.395	40.43
ALB	AL	AL	ALB	8	Albania	2740	3153731	150	39	20.068	41.143
ARM	AM	AM	ARM	51	Armenia	2820	3017661	142	145	44.563	40.534
AGO	AO	AO	AGO	24	Angola	124670	16095214	2	17	17.544	-12.296
ASM	AQ	AS	ASM	16	American Samoa	20	64051	9	61	-170.73	-14.318

The sp package


```
sp::spplot(wrld_simpl,"AREA")
```



The qtm command from the tmap package

Fast thematic map

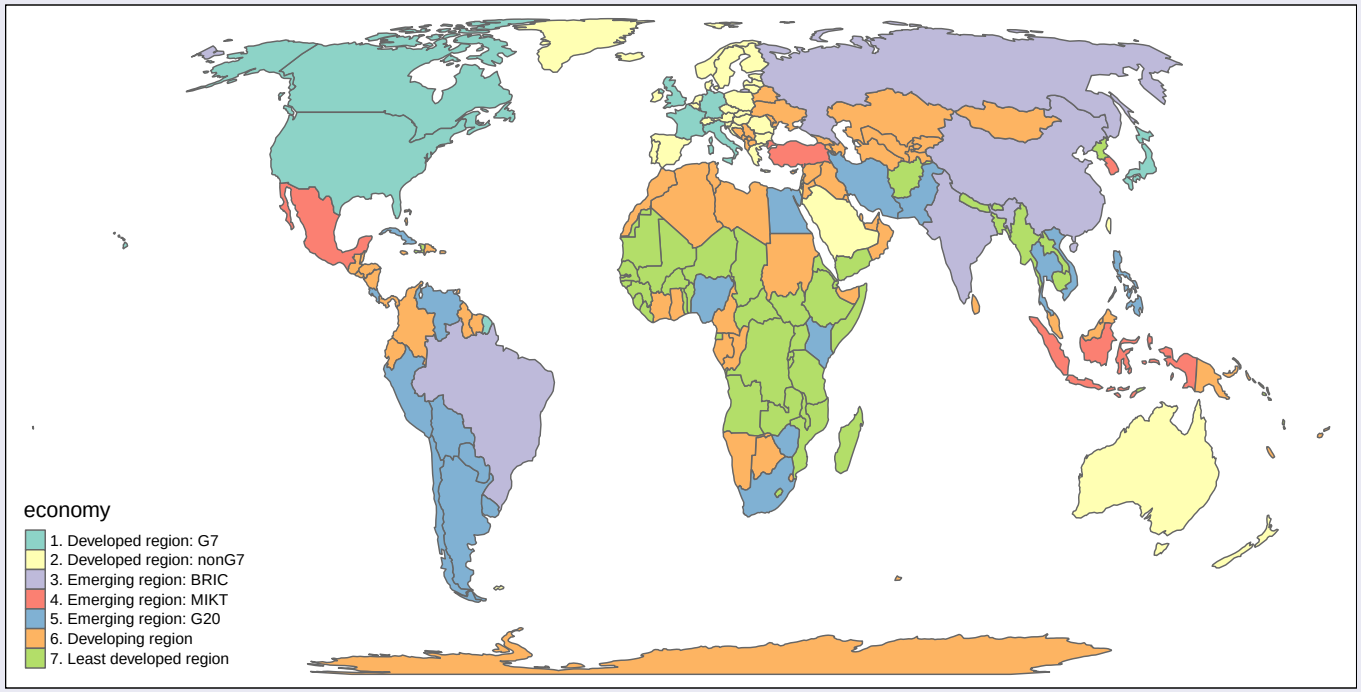
- With **qtm** you can create a fast thematic map
- Example from the **Vignette** for the tmap package

```
library(tmap)
data(World)
qtm(World)
```

To get more color in the map

Economic development status

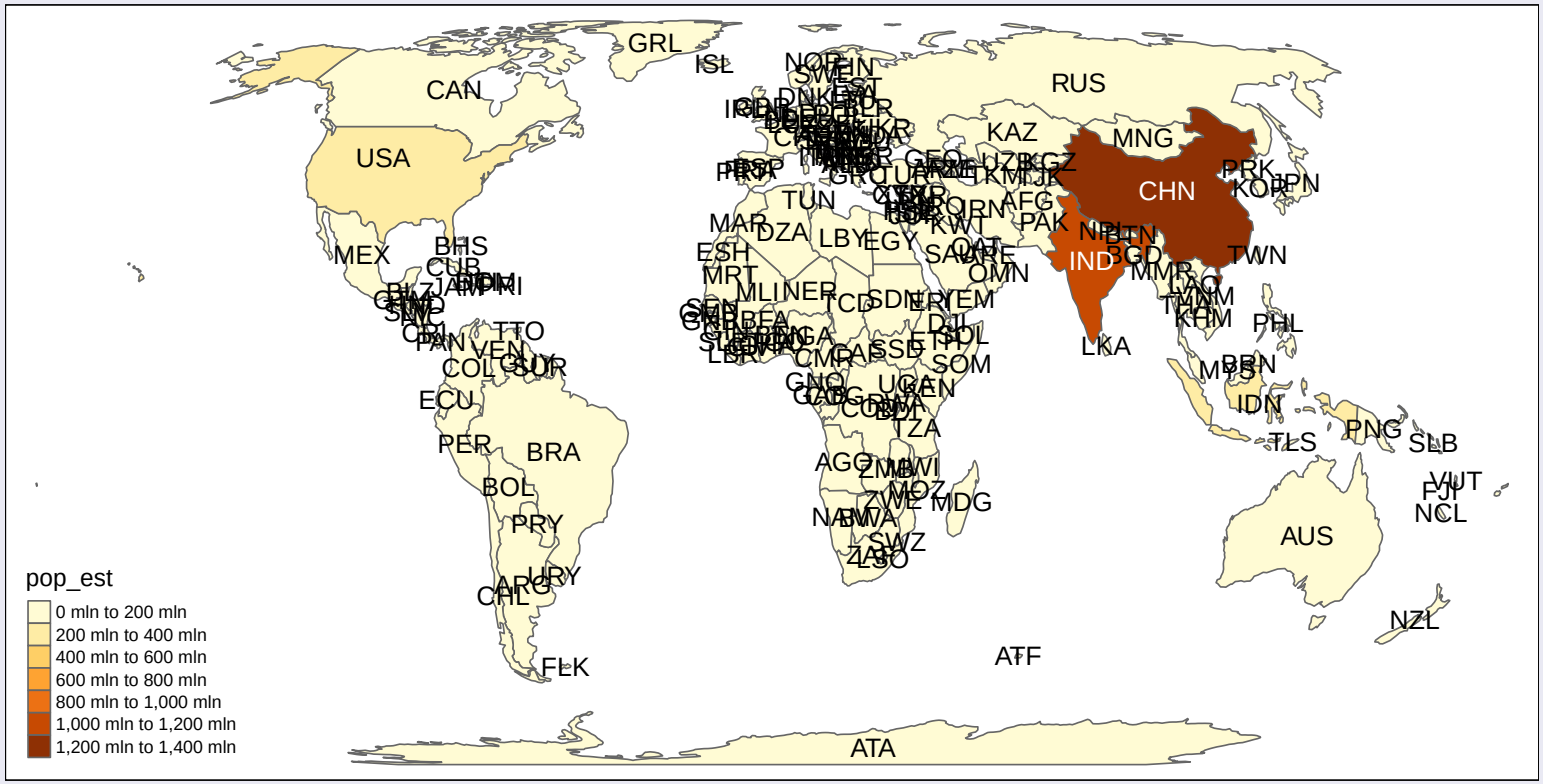
```
library(tmap)
data(World)
qtm(World, fill="economy")
```



A map with text

Population

```
qtm(World, fill="pop_est", text="iso_a3")
```



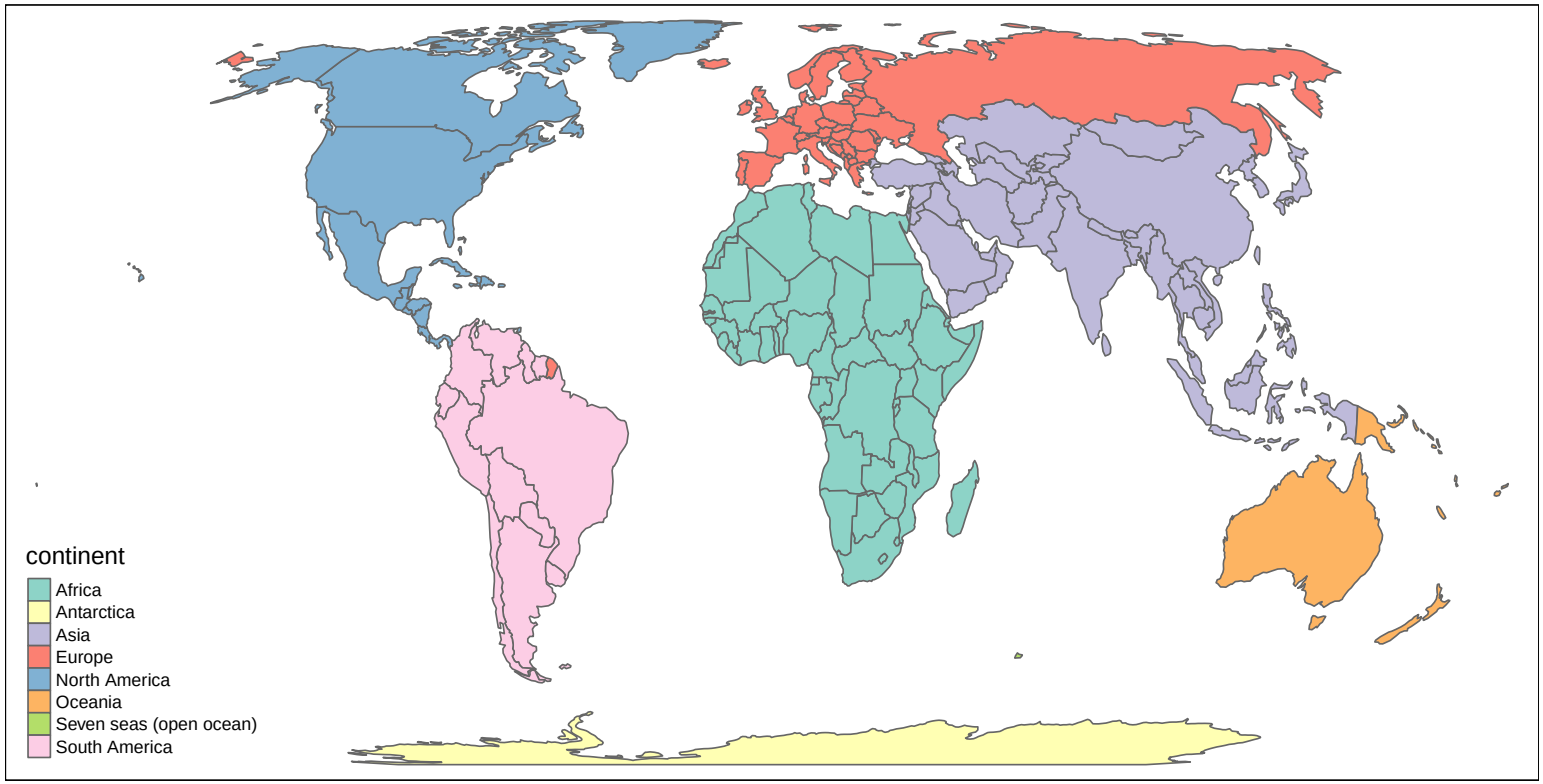
This Scheme is better:

GDP per capita

```
qtm(World, fill="gdp_cap_est", text="iso_a3",
    text.size="AREA")
```

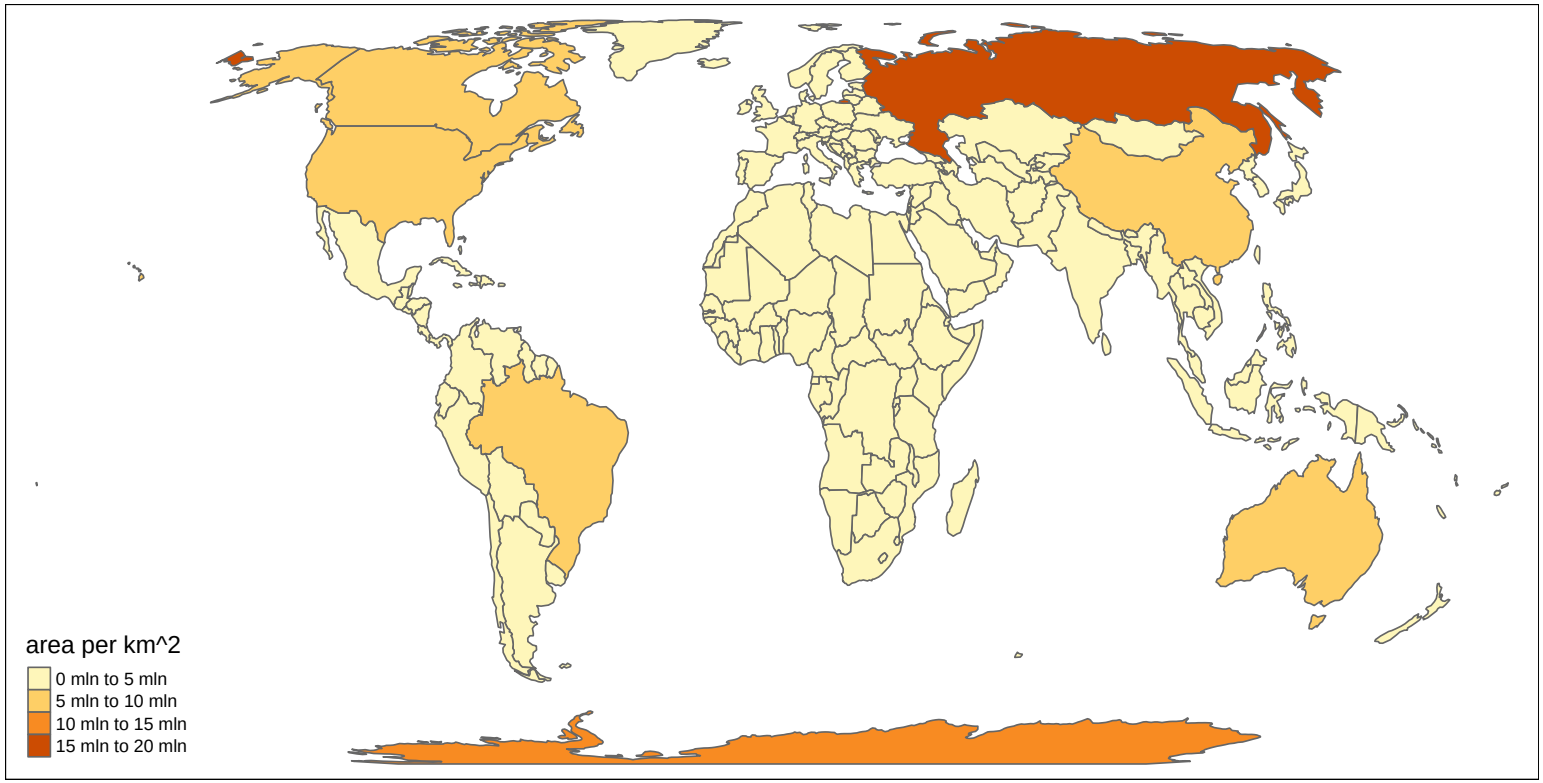
The variable continent


```
qtm(World, fill="continent")
```



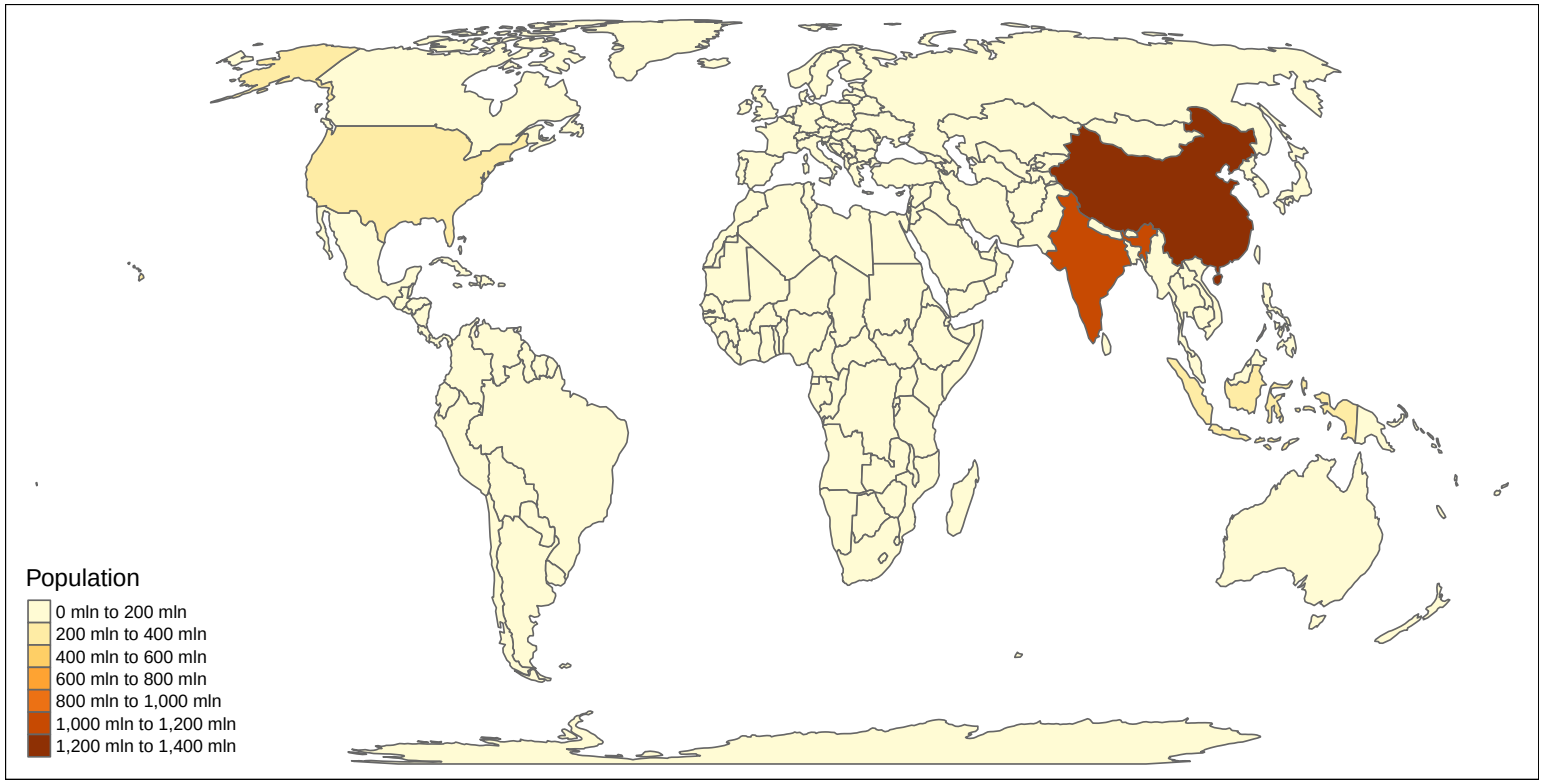
The variable area


```
qtm(World, fill="area") # Russia is huge
```



Population

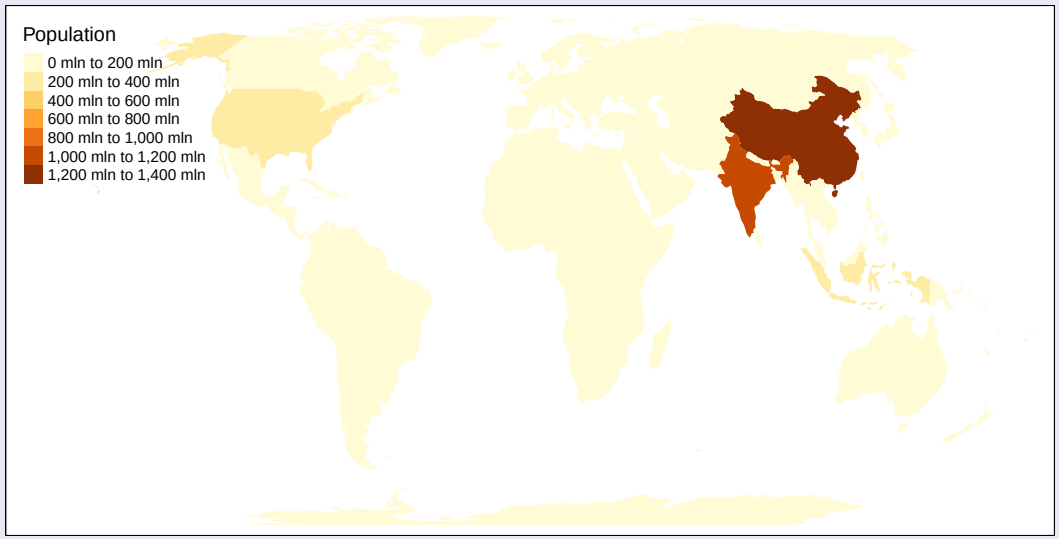

```
qtm(World, fill="pop_est",fill.title="Population")
```

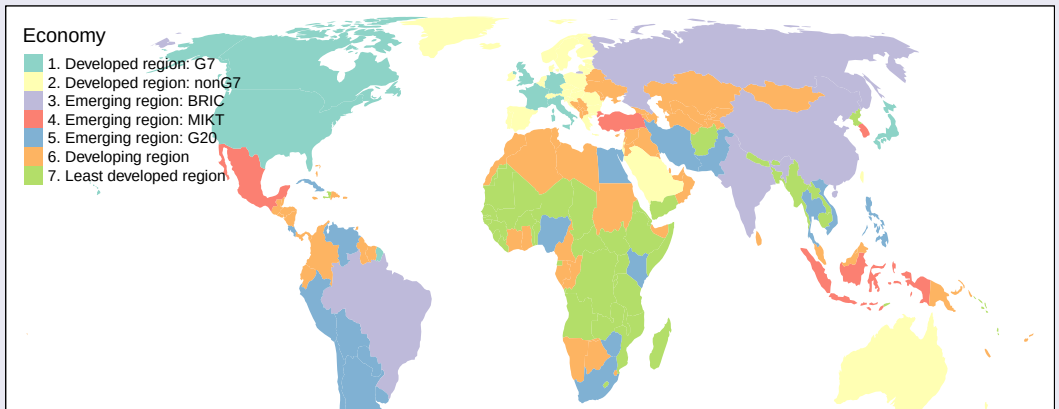


Two maps

Population and level of development

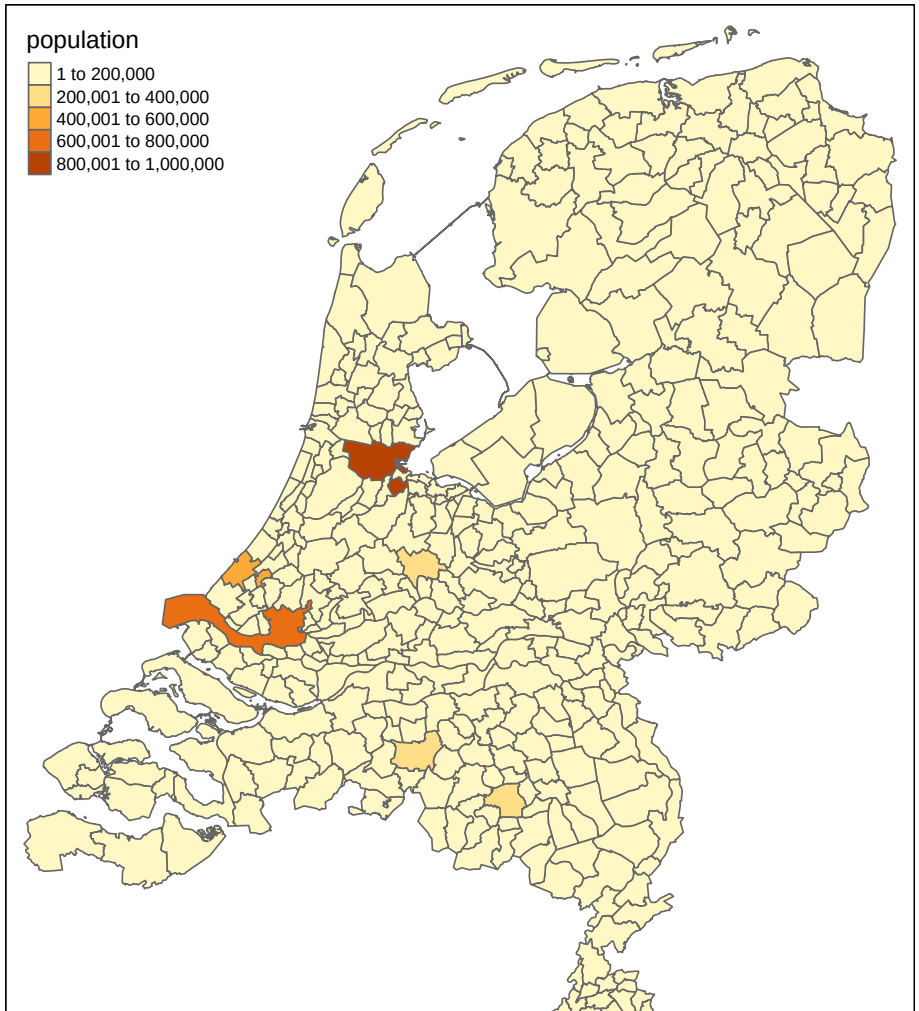
```
tm_shape(World) + tm_fill(c("pop_est", "economy"),
        title=c("Population", "Economy"))
```





Population in Dutch municipalities


```
data(NLD_muni)
qtm(NLD_muni,"population")
```

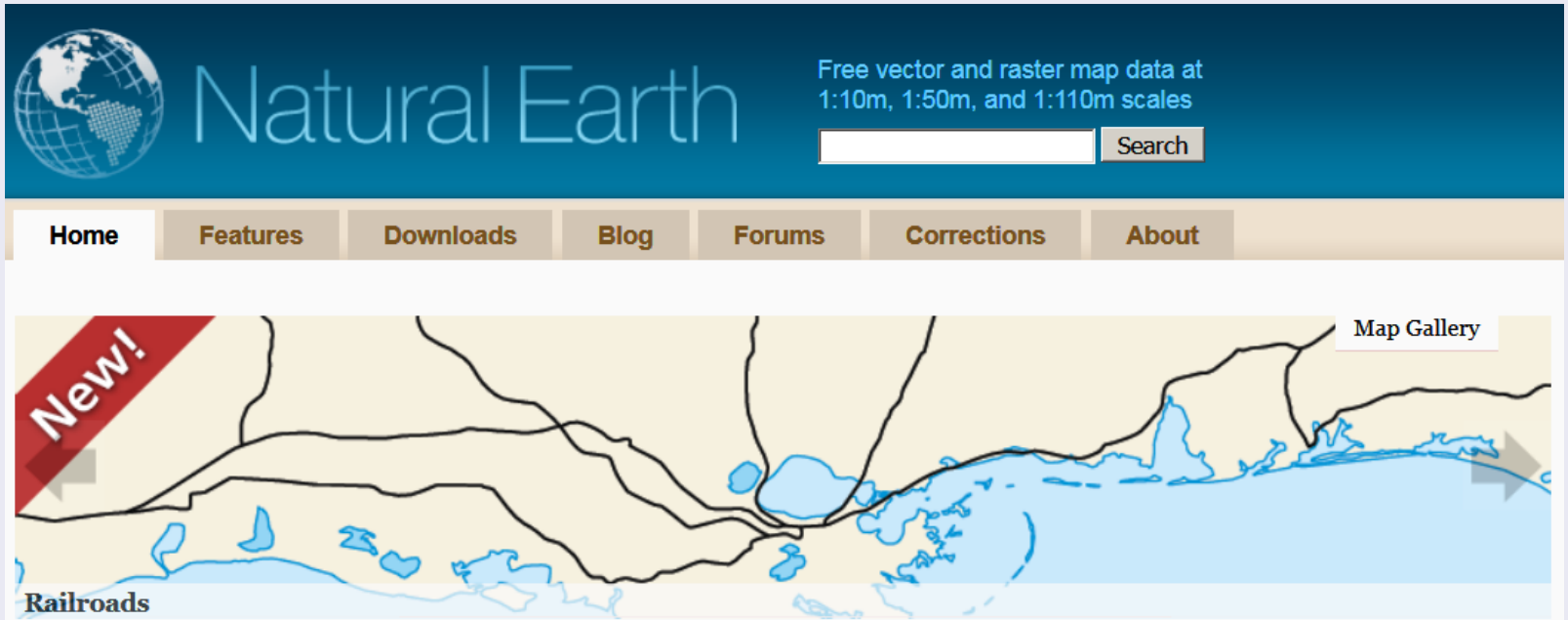


The World dataset

Natural Earth

- Dataset contains information from **Natural Earth**

```
library(tmap)
data(World)
```



Topics of the World dataset

Available variables in the data set

- **ISO classification**
- country name
- Area, population, population density,
- **Gross Domestic Product**
- Gross domestic product **at purchasing power parities**
- Economy, income group

Variables of the World Dataset

```
                   economy             income_grp gdp_cap_est life_exp well_being footprint inequality      HPI
1 7. Least developed region          5. Low income    784.1549    59.668        3.8      0.79  0.4265574 20.22535
2 7. Least developed region  3. Upper middle income   8617.6635        NA         NA        NA         NA       NA
3      6. Developing region  4. Lower middle income   5992.6588    77.347        5.5      2.21  0.1651337 36.76687
4      6. Developing region 2. High income: nonOECD  38407.9078        NA         NA        NA         NA       NA
                     geometry
1 MULTIPOLYGON (((5310471 451...
2 MULTIPOLYGON (((1531585 -77...
3 MULTIPOLYGON (((1729835 521...
4 MULTIPOLYGON (((4675864 313...
```

Simple Features


```
library(sf)
```

Simple Features


```
class(World)
```

```
## [1] "sf"         "data.frame"
```

- **Vignette of the `sf` package**

Attribute values and an abridged version of the geometry


```
## Simple feature collection with 100 features and 6 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
## precision:      double (default; no precision model)
## First 3 features:
##   BIR74 SID74 NWBIR74 BIR79 SID79 NWBIR79                              geom
## 1  1091     1      10  1364     0      19 MULTIPOLYGON(((-81.47275543...
## 2   487     0      10   542     3      12 MULTIPOLYGON(((-81.23989105...
## 3  3188     5     208  3616     6     260 MULTIPOLYGON(((-80.45634460...
```

Simple feature

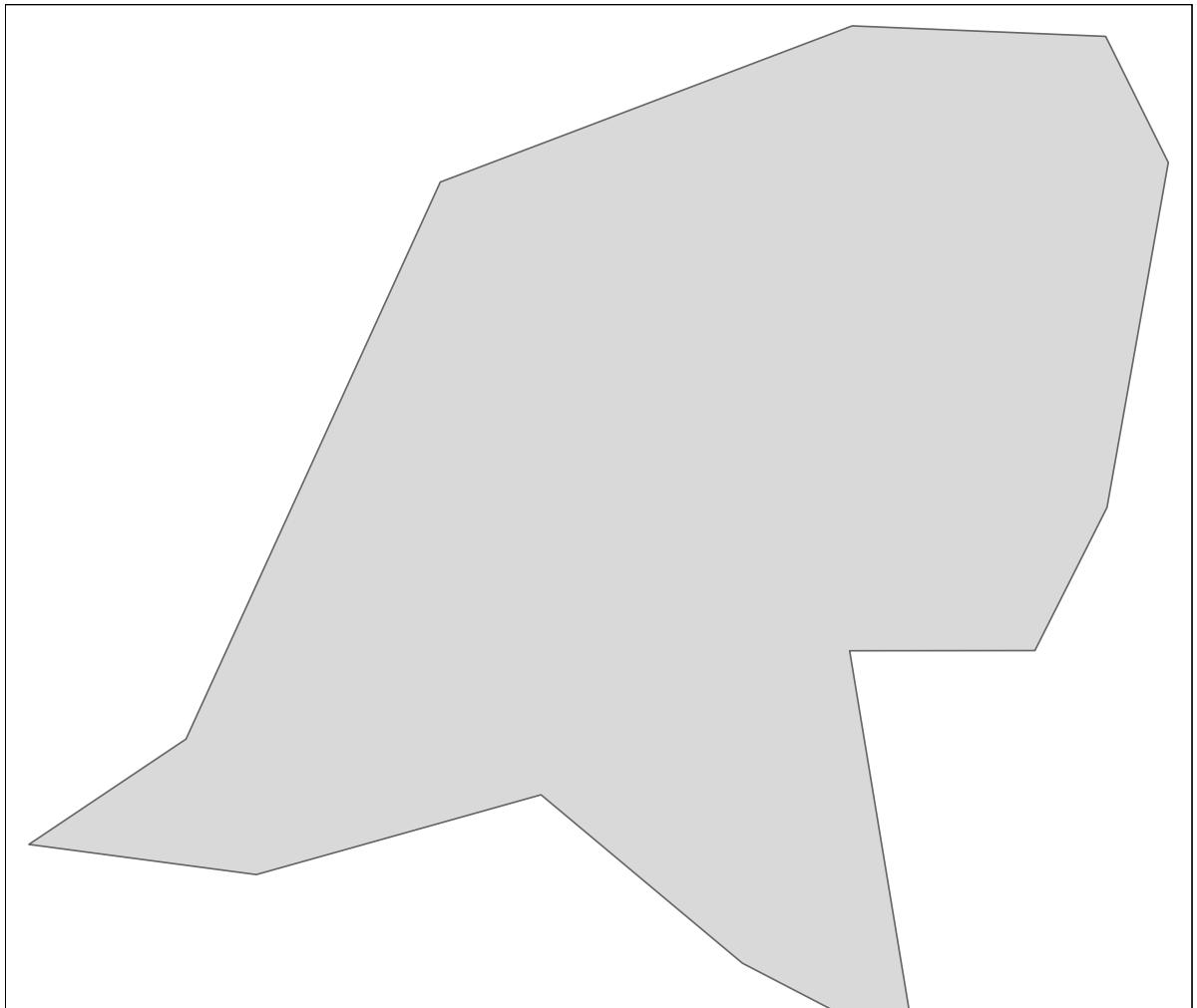
Simple feature geometry list–colum (sfc)

Simple feature geometry (sfg)

->

Use another country


```
tm_shape(World[World$name=="Netherlands", ]) +
    tm_polygons()
```



Eurostat Data


```
library(dplyr)
library(ggplot2)
```

Tools for Eurostat Open Data

```
library(eurostat)
```

```
# Download Geospatial Data from GISCO:
df60 <- get_eurostat_geospatial(resolution = 60)
# Same data - less detailed:
df1 <- get_eurostat_geospatial(resolution = 1)
```

The dataset

	id	CNTR_CODE	NUTS_NAME	LEVL_CODE	FID	NUTS_ID	geometry	geo
1	BG	BG	БЪЛГАРИЯ	0	BG	BG	[object Object]	BG
2	CH	CH	SCHWEIZ/SUISSE/SVIZZERA	0	CH	CH	[object Object]	CH
3	CY	CY	ΚΥΠΡΟΣ	0	CY	CY	[object Object]	CY

A more detailed map of the Netherlands


```
plot(df60[df60$LEVL_CODE == 0 & df60$CNTR_CODE == "NL
```

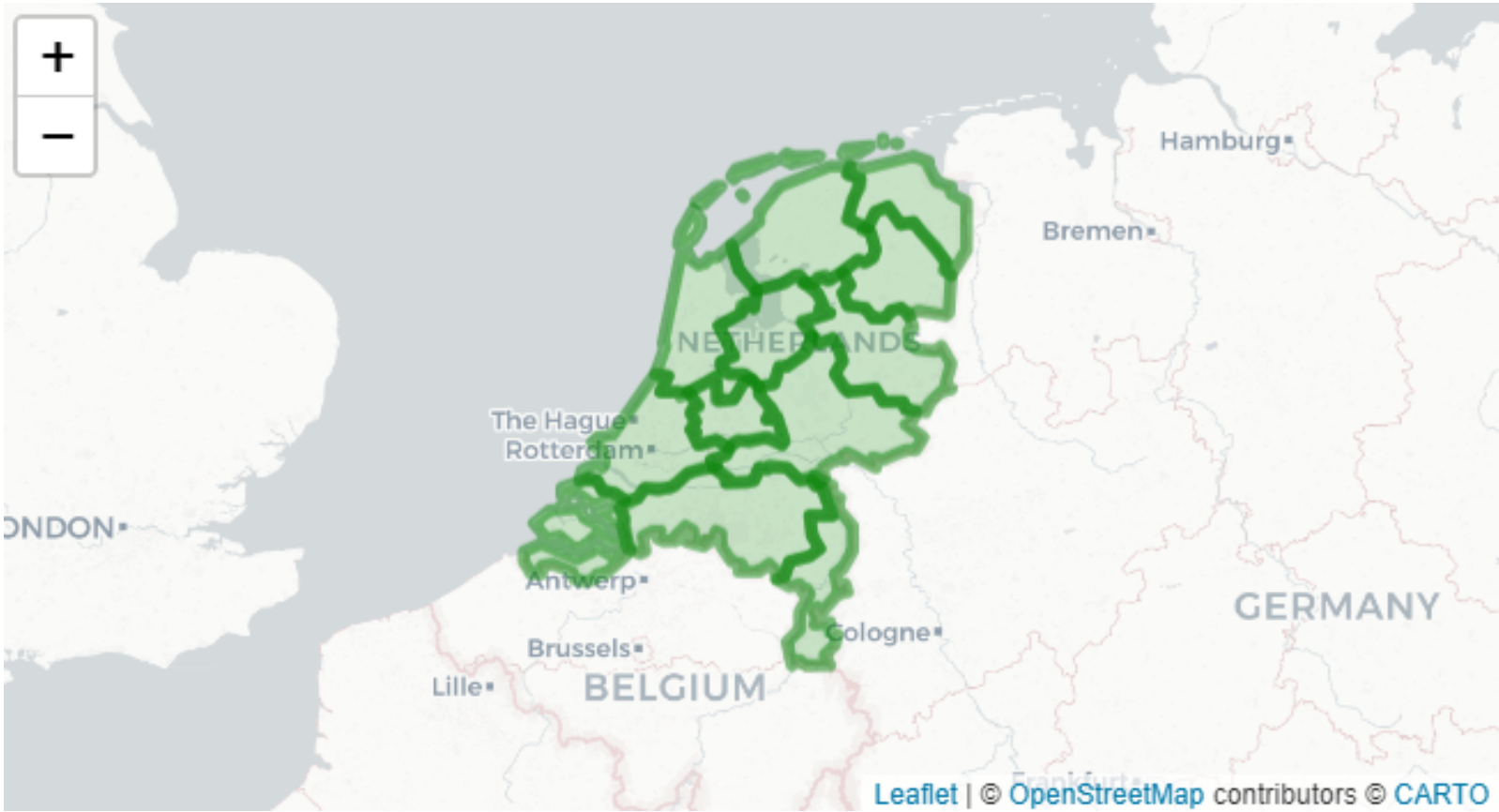
NUTS2 of the Netherland


```
CE.sf <- df60 %>%
  filter(LEVL_CODE == 2 &
  CNTR_CODE %in% c("NL")) %>%
  select(NUTS_ID)
plot(CE.sf,col=1:12)
```

- NUTS 3 is the limit

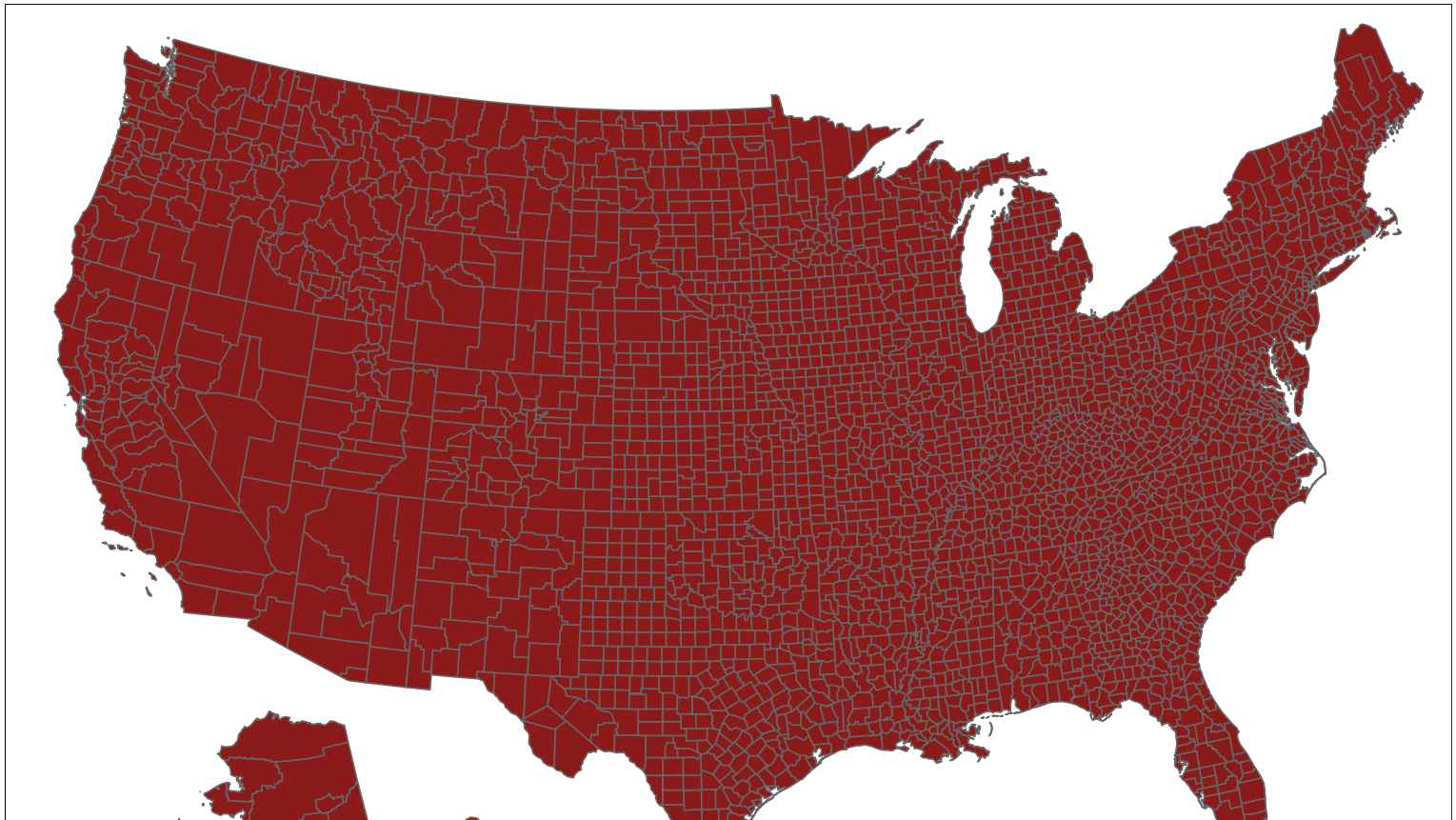
An interactive map of the Netherlands


```
library(leaflet)
leaflet(CE.sf) %>%
  addProviderTiles("CartoDB.Positron") %>%
  addPolygons(color = "green")
```



The package tidycensus

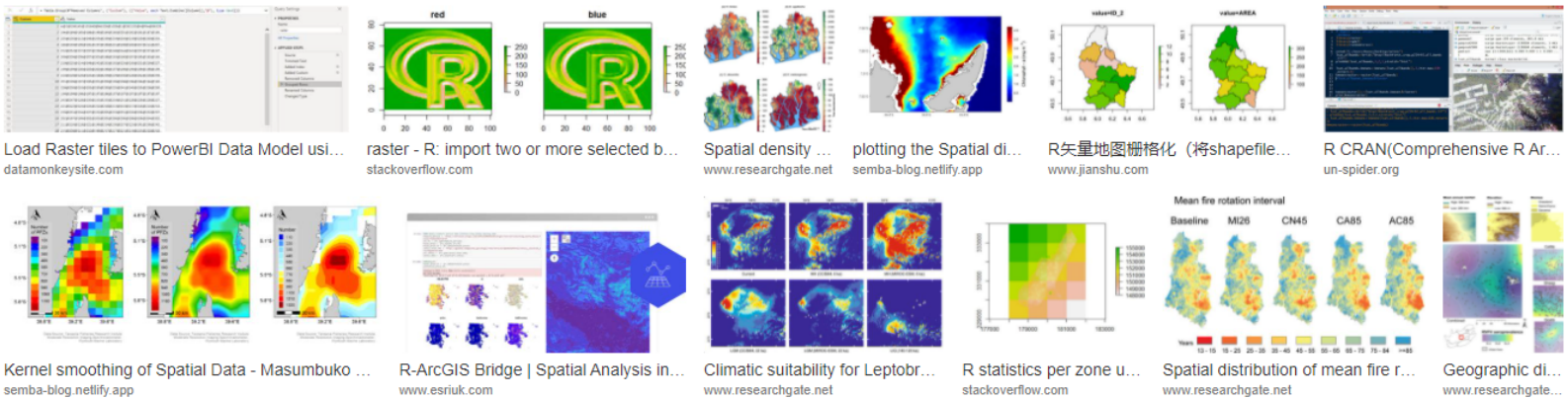

```
library(tidycensus)
data(county_laea)

qtm(county_laea,fill=c("#8B1A1A"))
```



raster - Geographic Data Analysis and Modeling

Quick high quality maps with R

Jan-Philipp Kolb



Description:

Reading, writing, manipulating, analyzing and modeling of spatial data. The package implements basic and high-level functions for raster data and for vector data operations such as intersections.

Author and contributors:

Robert J. Hijmans [cre, aut], Jacob van Etten [ctb], Michael Sumner [ctb], Joe Cheng [ctb], Dan Baston [ctb], Andrew Bevan [ctb], Roger Bivand [ctb], and many more

Global Administrative Boundaries

Get the data

```
FRA4 <- raster::getData('GADM', country='FRA',
                                level=4)
```

Overview of the data

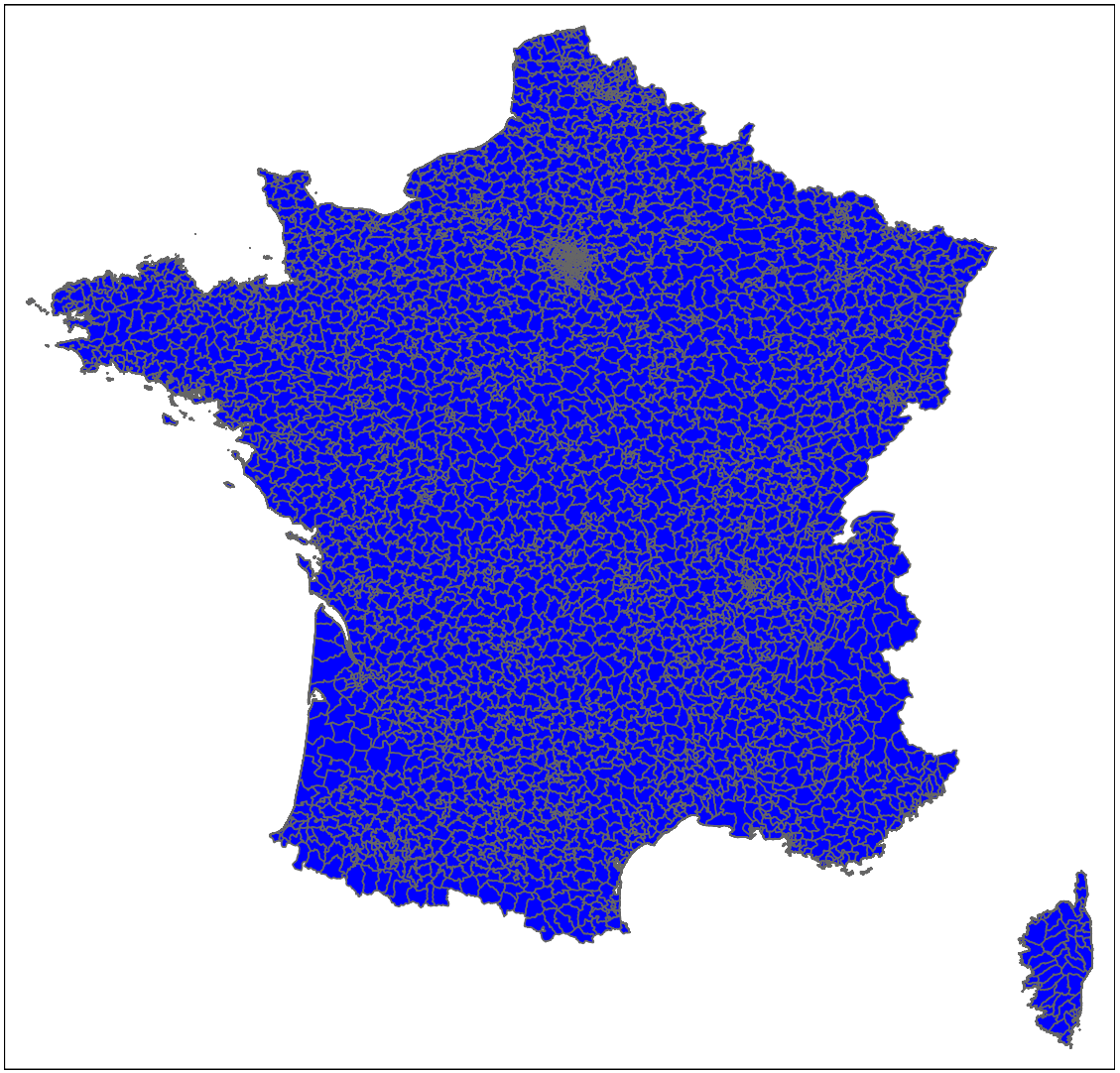
RPubs by RStudio

Show 10 entries Search:

	GID_0	NAME_0	GID_1	NAME_1	GID_2	NAME_2	GID_3	NAME_3	GID_4	NAME_4
1	FRA	France	FRA.1_1	Auvergne-Rhône-Alpes	FRA.1.1_1	Ain	FRA.1.1.1_1	Belley	FRA.1.1.1.1_1	Ambérieu-en-Bugey
2	FRA	France	FRA.1_1	Auvergne-Rhône-Alpes	FRA.1.1_1	Ain	FRA.1.1.1_1	Belley	FRA.1.1.1.2_1	Belley
3	FRA	France	FRA.1_1	Auvergne-Rhône-Alpes	FRA.1.1_1	Ain	FRA.1.1.1_1	Belley	FRA.1.1.1.3_1	Champagne-en-Valromey
4	FRA	France	FRA.1_1	Auvergne-Rhône-Alpes	FRA.1.1_1	Ain	FRA.1.1.1_1	Belley	FRA.1.1.1.4_1	Hauteville-Lompnes
5	FRA	France	FRA.1_1	Auvergne-Rhône-Alpes	FRA.1.1_1	Ain	FRA.1.1.1_1	Belley	FRA.1.1.1.5_1	Lagnieu
6	FRA	France	FRA.1_1	Auvergne-Rhône-	FRA.1.1_1	Ain	FRA.1.1.1_1	Belley	FRA.1.1.1.6_1	Lhuis

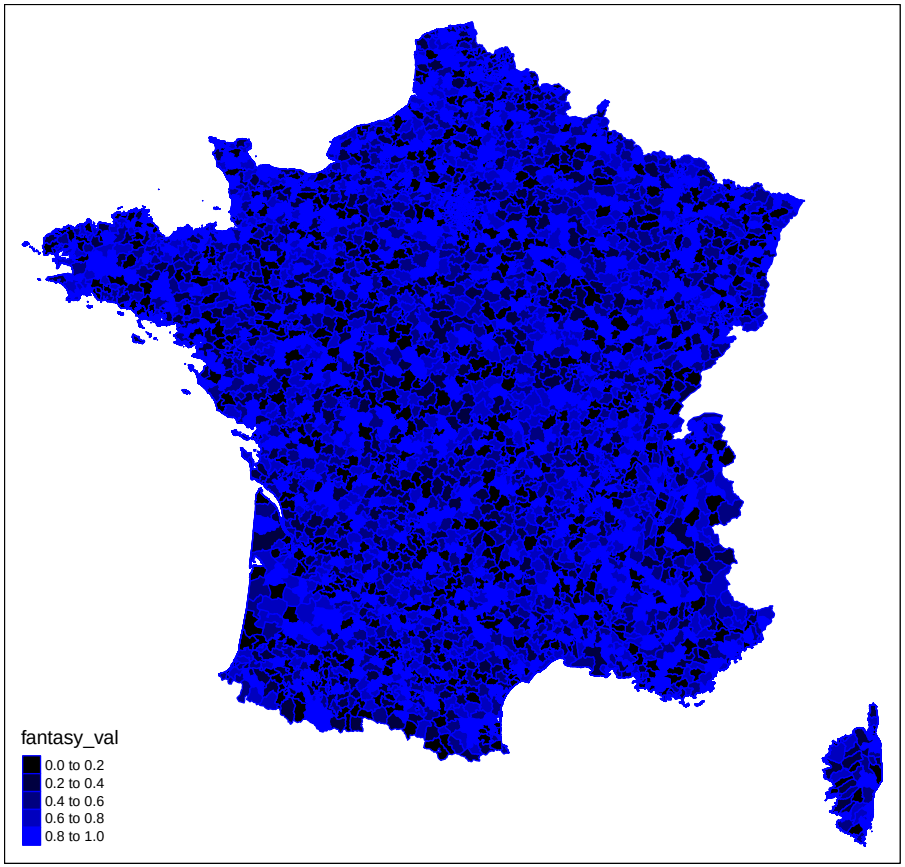
Plot the map


```
qtm(FRA4,fill="blue")
```

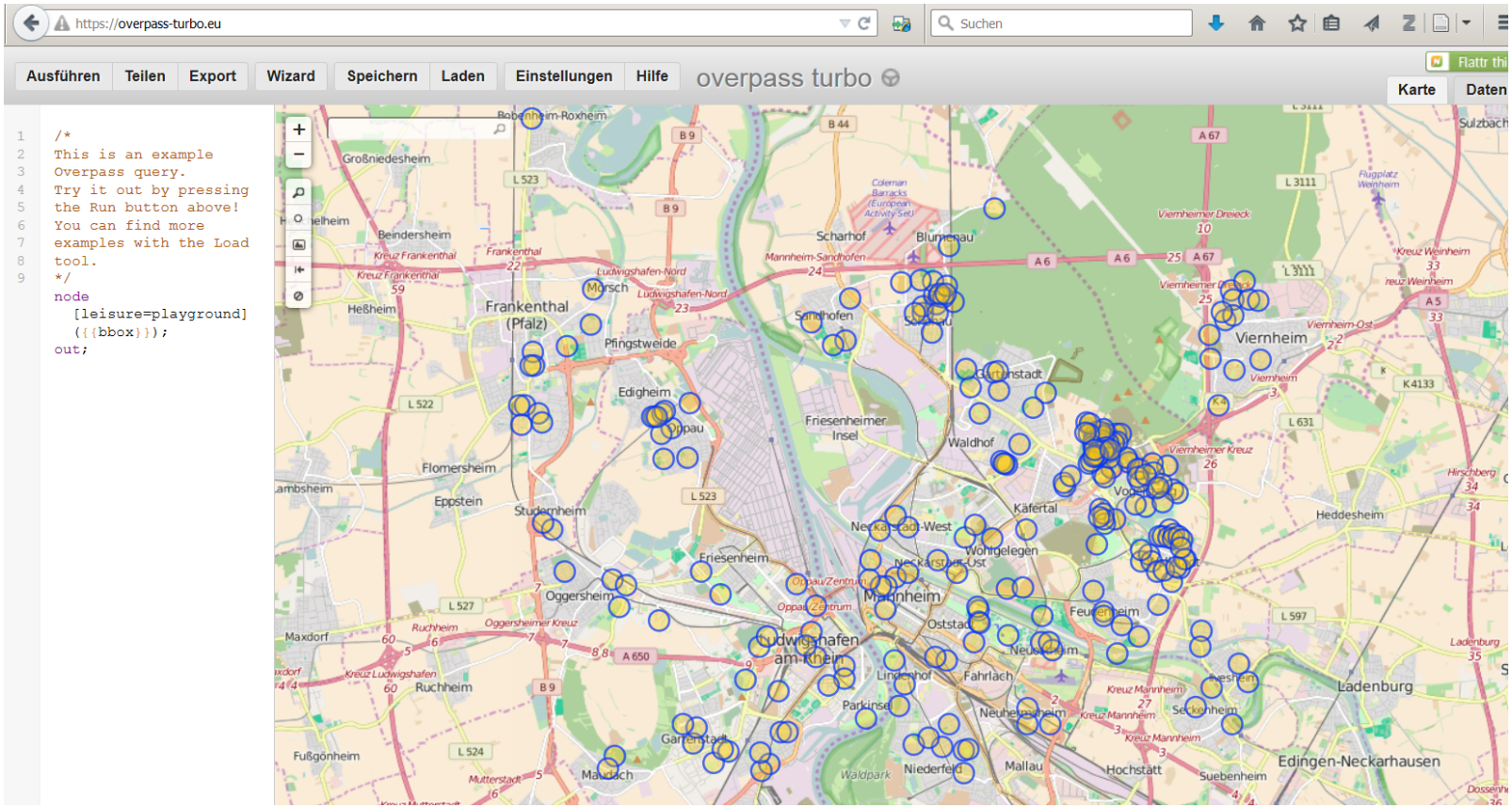


Create colour gradiation

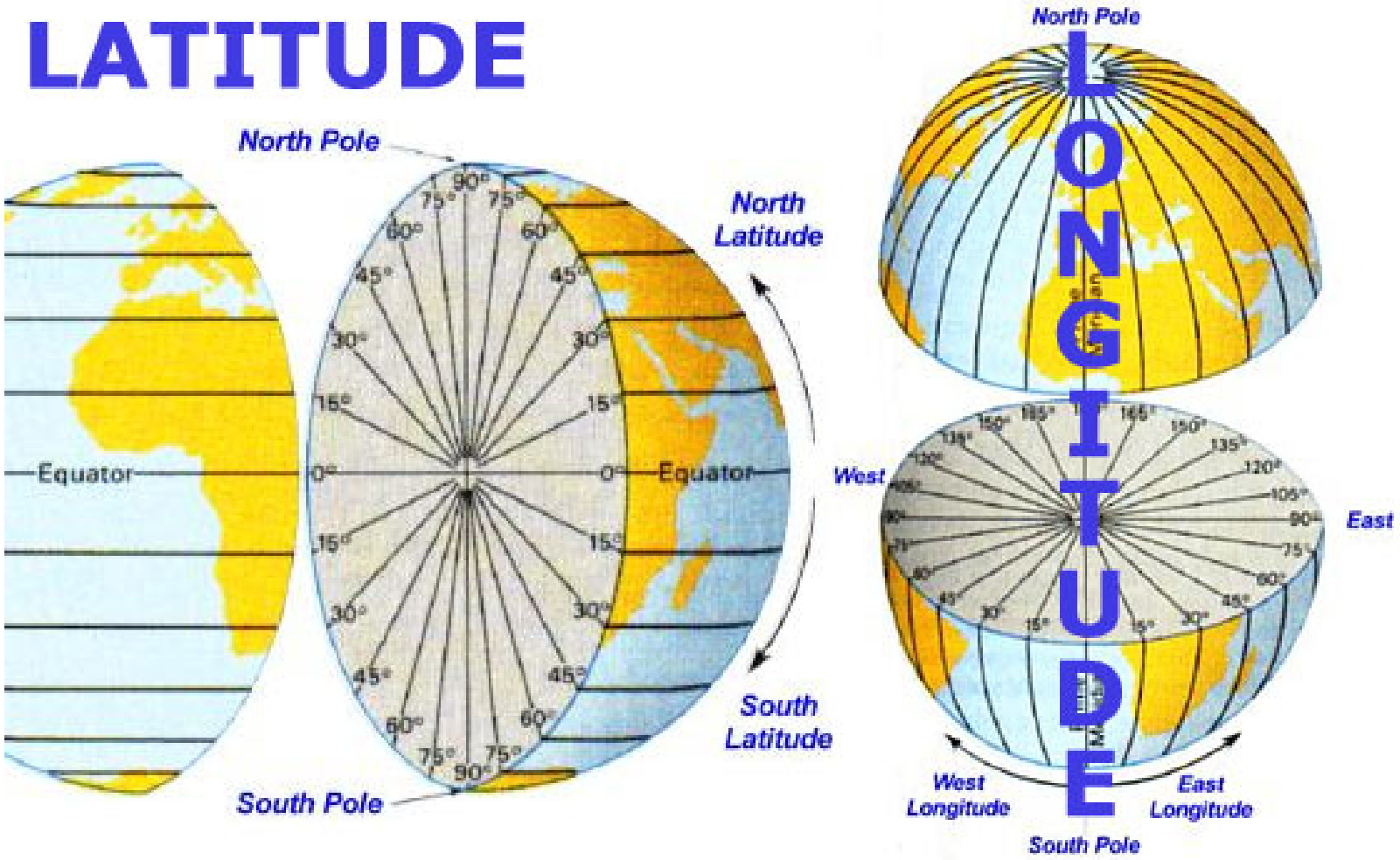

```
FRA4$fantasy_val <- runif(nrow(FRA4))
  qtm(FRA4,"fantasy_val",
      fill.palette = rgb(0,0,seq(0,1,.1),0),
      borders = "blue")
```



Example: Overpass API



Longitude and Latitude



tmaptools - Thematic Map Tools

```
library(tmaptools)
```

Author: Martijn Tennekes

```
citation("tmaptools")
```

Description

Set of tools for reading and processing spatial data. The aim is to supply the workflow to create thematic maps. This package also facilitates 'tmap', the package for visualizing thematic maps.

Geocoordinates


```
(gc_z <- geocode_OSM("Zürich"))
```

```
## $query
## [1] "Zürich"
##
## $coords
##          x          y
##  8.541042 47.374449
##
## $bbox
##      xmin       ymin       xmax       ymax
##  8.448006 47.320220  8.625441 47.434666
```

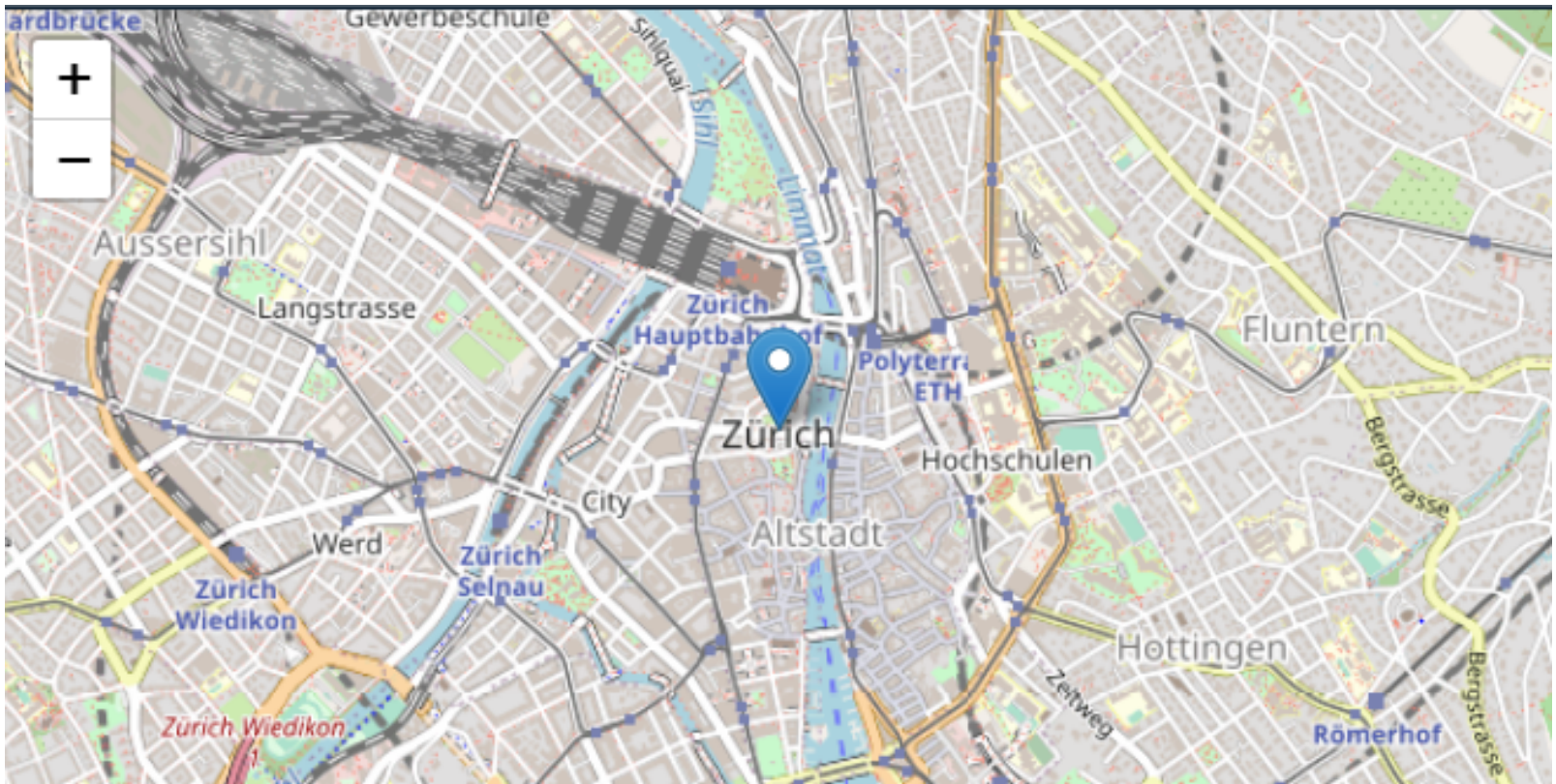
Get more information


```
gc_z <- geocode_OSM("Zürich",details = T)
names(gc_z)
```

```
##  [1] "query"        "coords"       "bbox"
##  [6] "osm_id"       "place_rank"   "display_name"
## [11] "importance"   "icon"
```

An interactive map with a popup


```
library(leaflet)
gcz <- geocode_OSM("Zürich")
leaflet() %>%  addTiles() %>%
  addMarkers(lng=gcz$coords[1] , lat=gcz$coords[2] ,
             popup="The conference place")
```

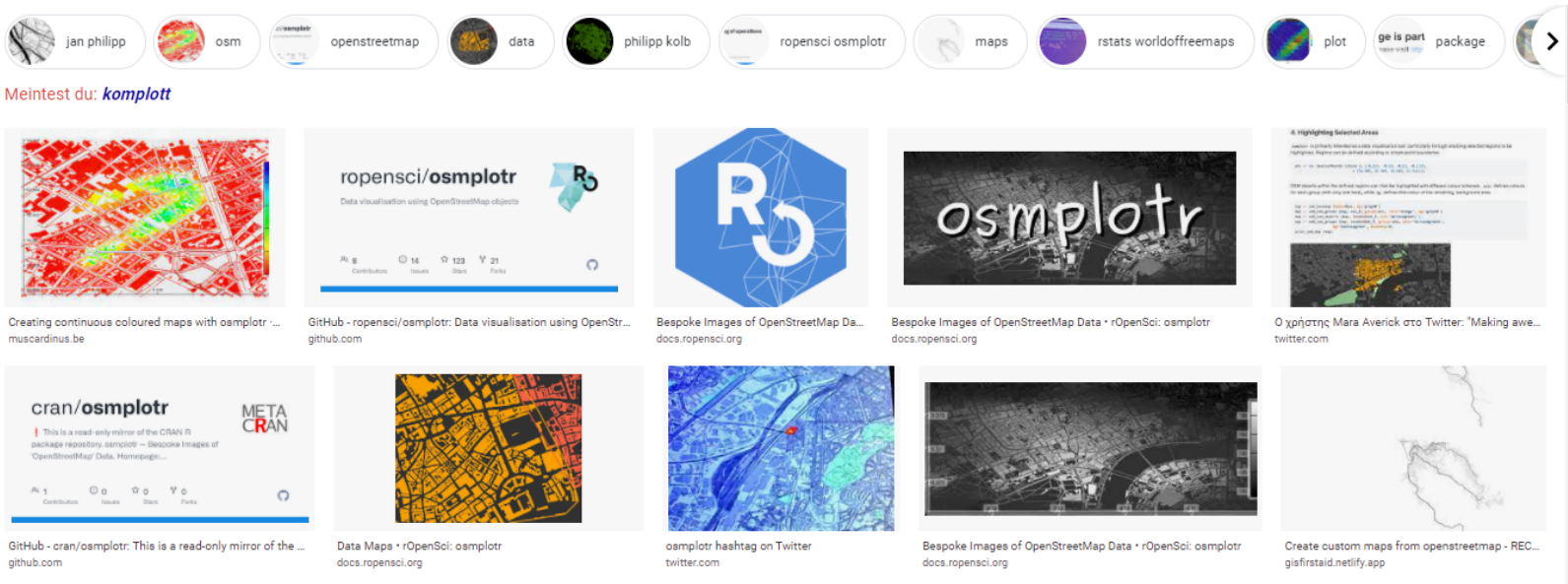


A package to get Openstreetmap data

Author: Mark Padgham

```
library(osmplotr)
```

```
citation("osmplotr")
```

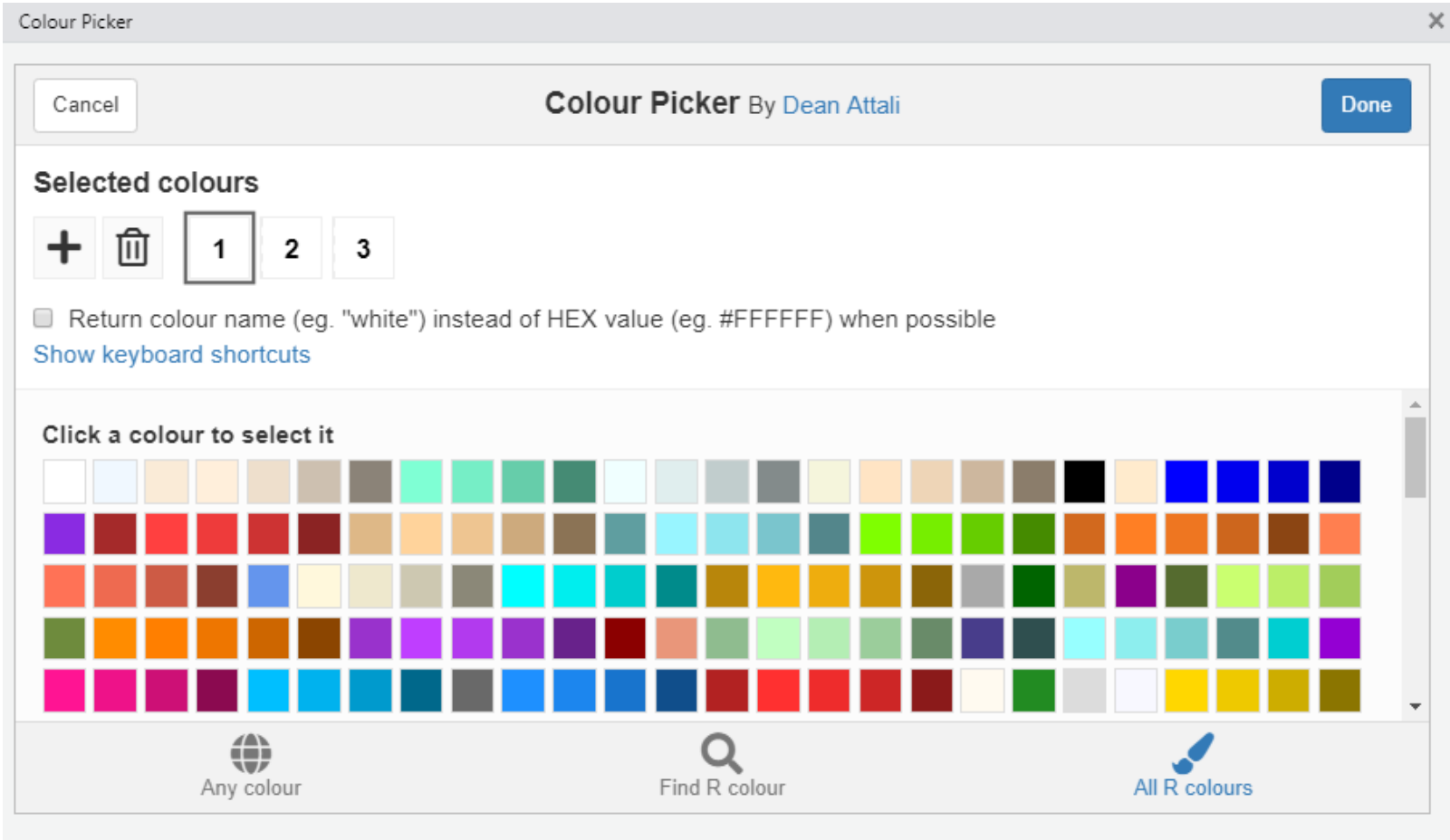


Buildings within a bounding box


```
bbox <- get_bbox (c(8.4539 , 49.4805  ,
                    8.4774 , 49.4943 ))
dat_M <- extract_osm_objects (key = 'building',
                              bbox = bbox)
```

```
qtm(dat_M,fill=c("purple"),borders="black")
```

Colour picker



The package osmdata

Install and load osmdata

```
install.packages("osmdata")
```

```
library(osmdata)
```

Get a bounding box for a city

```
bbox <- getbb("Berlin")
```

Streets of Berlin

OSM Map Feature key=highway value=primary

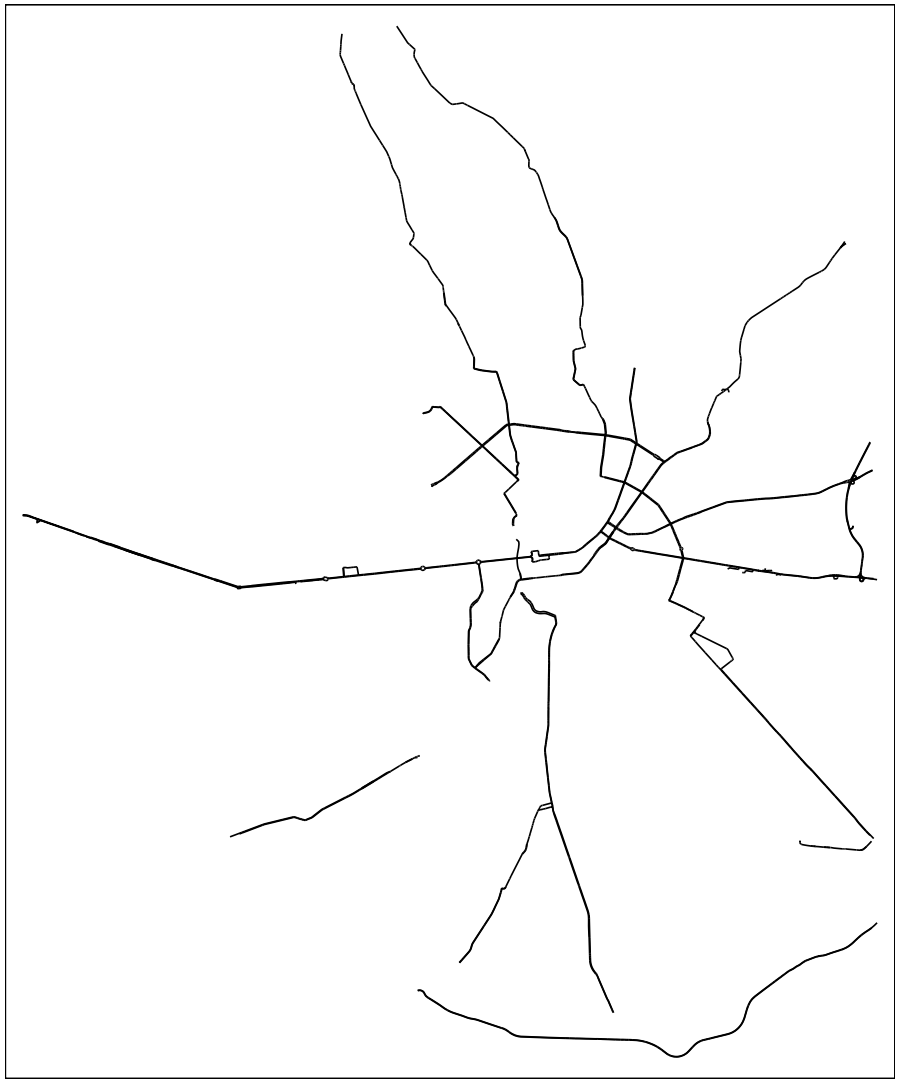
highway	primary		The next most important roads in a country's system. (Often link larger towns.)		

Get data with package osmdata

```
dat <- extract_osm_objects(key = 'highway',
                           value = "primary",
                           bbox = bbox)
```

A quick map for the primary streets in Berlin


```
qtm(dat)
```



Get data for secondary roads in Berlin

OSM map feature

highway	secondary		The next most important roads in a country's system. (Often link towns.)		

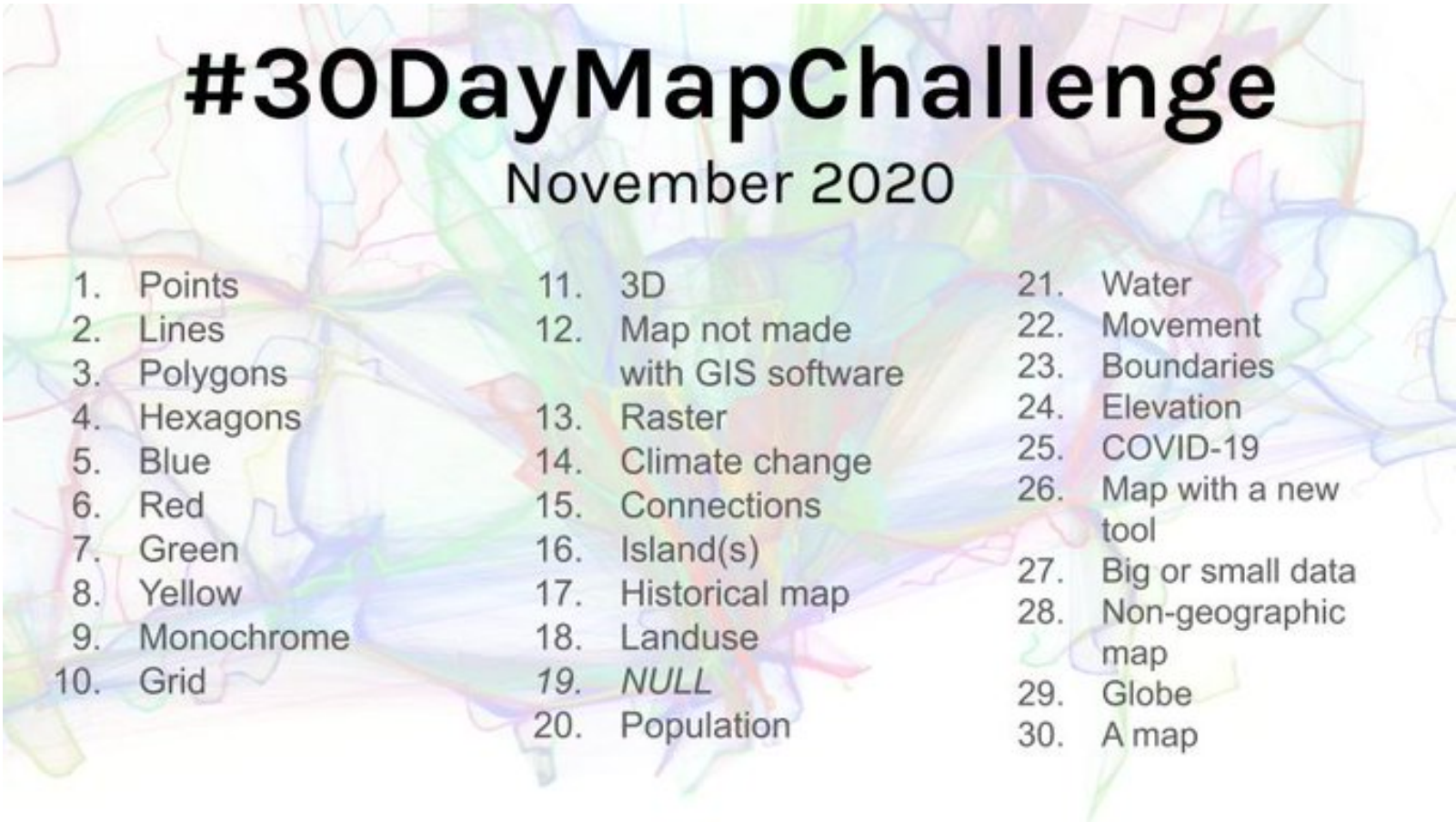
New Information - same bounding box

```
dat_s <- extract_osm_objects(key = 'highway',
                             value = "secondary",
                             bbox = bbox)
```

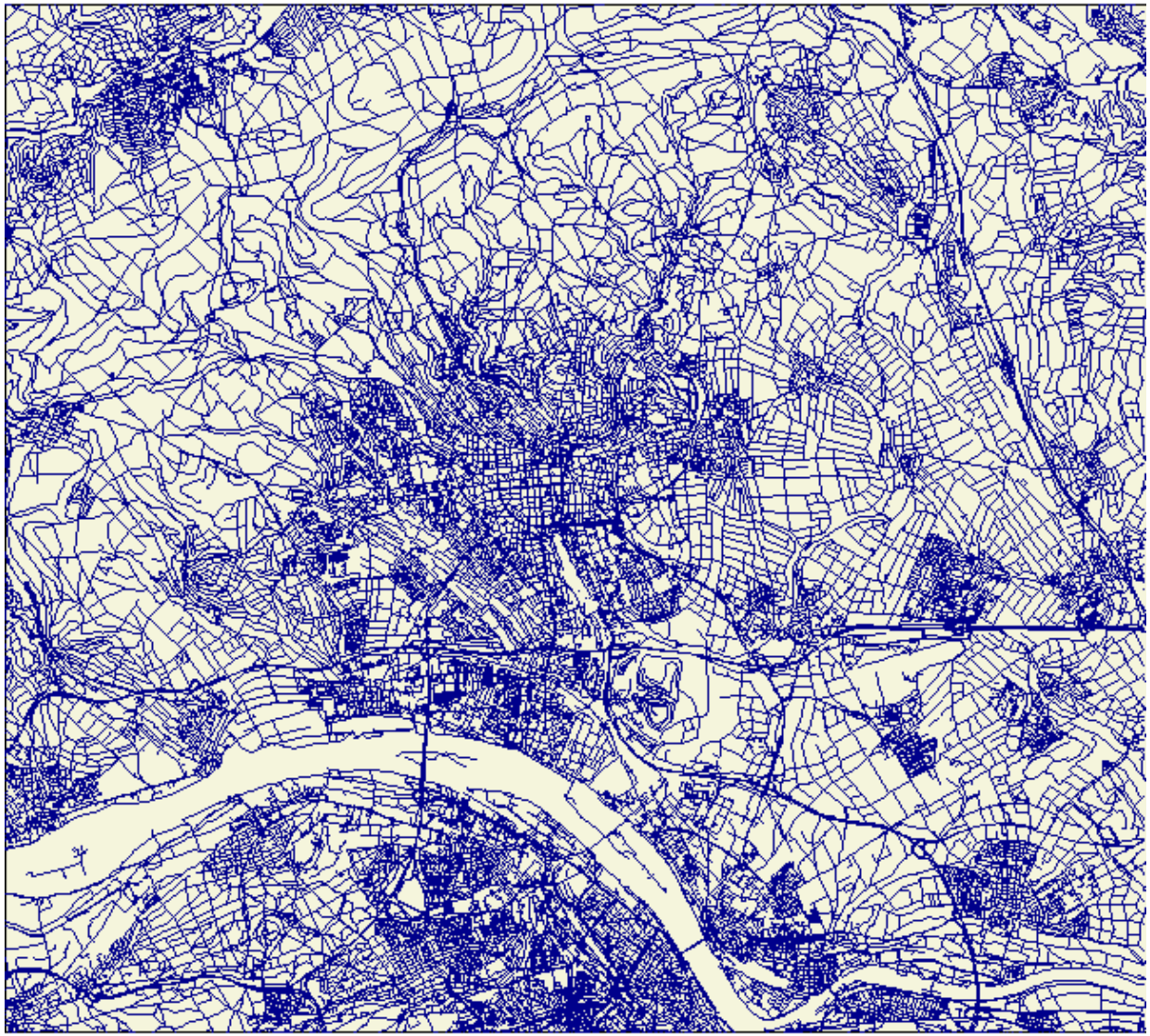
Plot the map


```
map <- osm_basemap(bbox = bbox, bg = "#F5F5DC")
map <- add_osm_objects(map, dat, col = "#00008B")
map <- add_osm_objects(map, dat_s, col = "purple")
print_osm_map(map)
```

30daymapchallenge



Day 9 monochrome



Load example data

Data source Eurostat

- Data about unemployment in Europe

```
url <- "https://raw.githubusercontent.com/Japhilko/
GeoData/master/2015/data/Unemployment07a13.csv"

Unemp <- read.csv(url)
```

Excursus: the command `match`

Create two example vectors

```
vec_a <- c("A",2,6,1,"C")
vec_b <- c(1,"C",2)
```

Bringing the two vectors together

- With the function `match` you can see which element of the first vector matches the second vector.

```
match(vec_a,vec_b)
```

```
## [1] NA  3 NA  1  2
```

Use the package tmap with your data

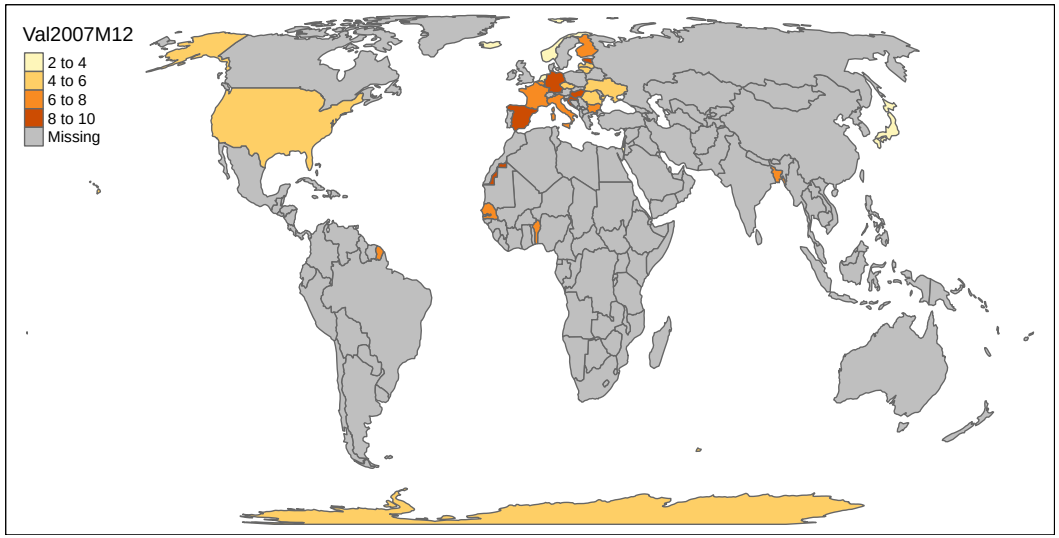

```
library("tmap")
```

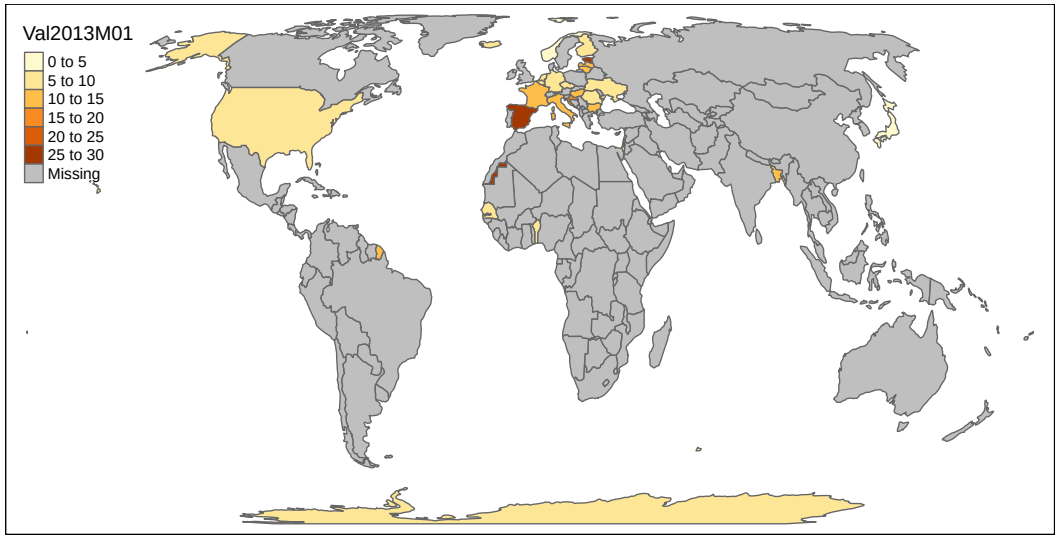
Match the data

```
iso_a2<- substr(World$iso_a3,1,2)
ind <- match(iso_a2,Unemp$GEO)
World$Val2007M12 <- Unemp$Val2007M12[ind]
World$Val2013M01 <- Unemp$Val2013M01[ind]
```

Plot a map

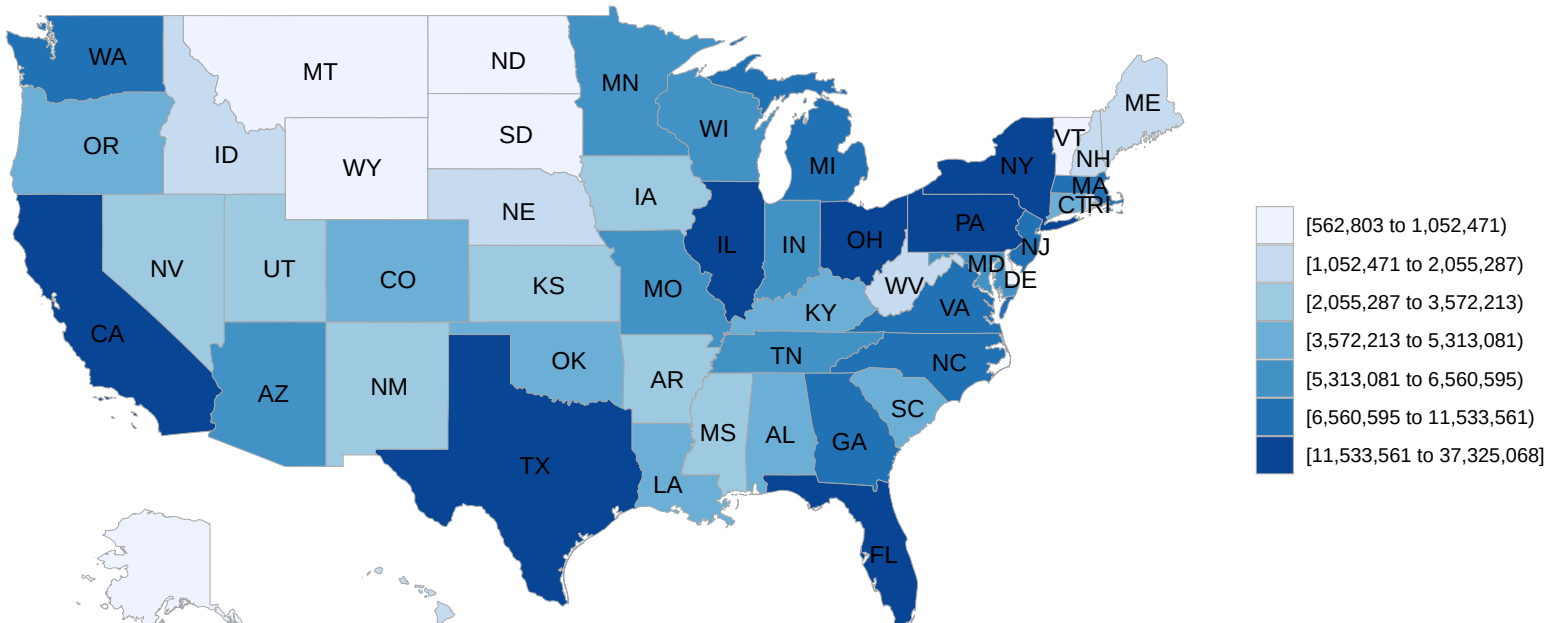

```
qtm(World,c("Val2007M12","Val2013M01"))
```





The package choroplethrMaps


```
library("choroplethr")
library("choroplethrMaps")
data(df_pop_state)
state_choropleth(df_pop_state)
```



OSM Map features

Amenity

Used to map facilities used by visitors and residents. For example: toilets, telephones, banks, pharmacies, cafes, parking and schools. See the page **Amenities** for an introduction on its usage.

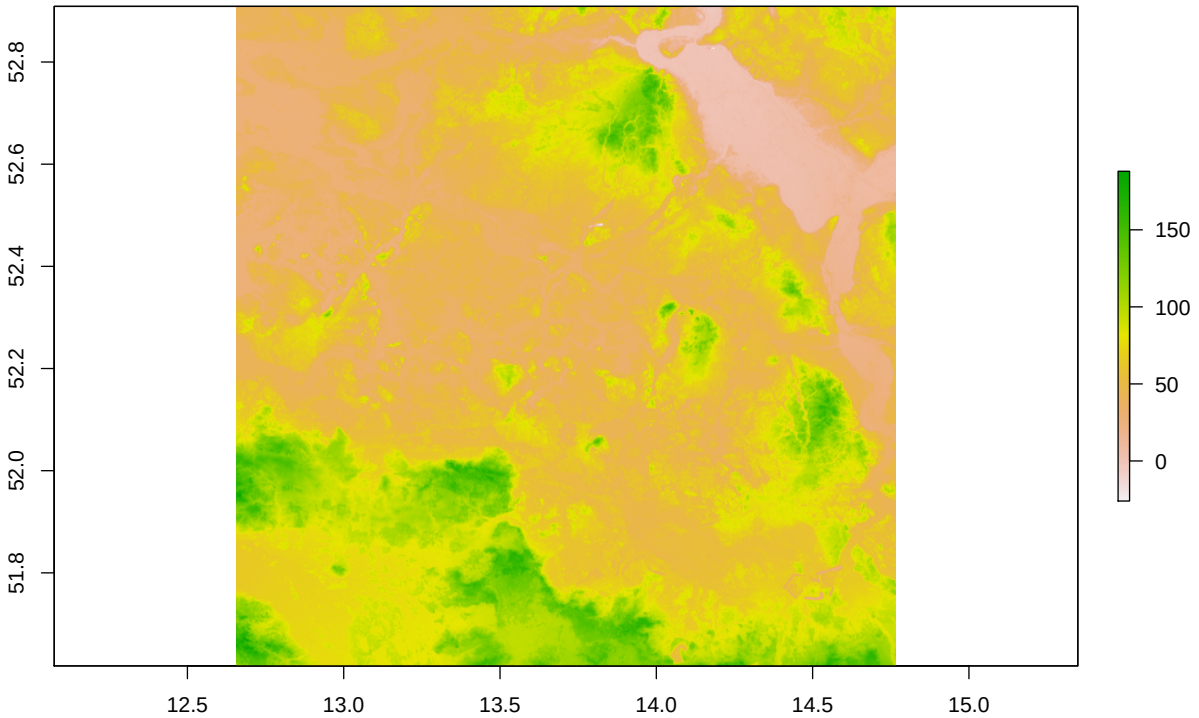
Key	Value	Element	Comment	carto-Rendering	Photo
Sustenance					
amenity	bar		**Bar** is a purpose-built commercial establishment that sells alcoholic drinks to be consumed on the premises. They are characterised by a noisy and vibrant atmosphere, similar to a party and usually don't sell food. See also the description of the tags `amenity=pub;bar;restaurant` for a distinction between these.		
amenity	biergarten		**Biergarten** or **beer garden** is an open-air area where alcoholic beverages along with food is prepared and served. See also the description of the tags `amenity=pub;bar;restaurant`. A biergarten can commonly be found attached to a beer hall, pub, bar, or restaurant. In this case, you can use `biergarten=yes` additional to `amenity=pub;bar;restaurant`.		

Get more data from OSM


```
bars <- extract_osm_objects(key = 'amenity',
                            #value = "bar",
                            bbox = bbox)
```

The elevatr package


```
library(elevatr)
# load("data/dat_osm_barcelona_metro.Rdata")
elevation <- get_elev_raster(dat, z = 9)
plot(elevation)
```



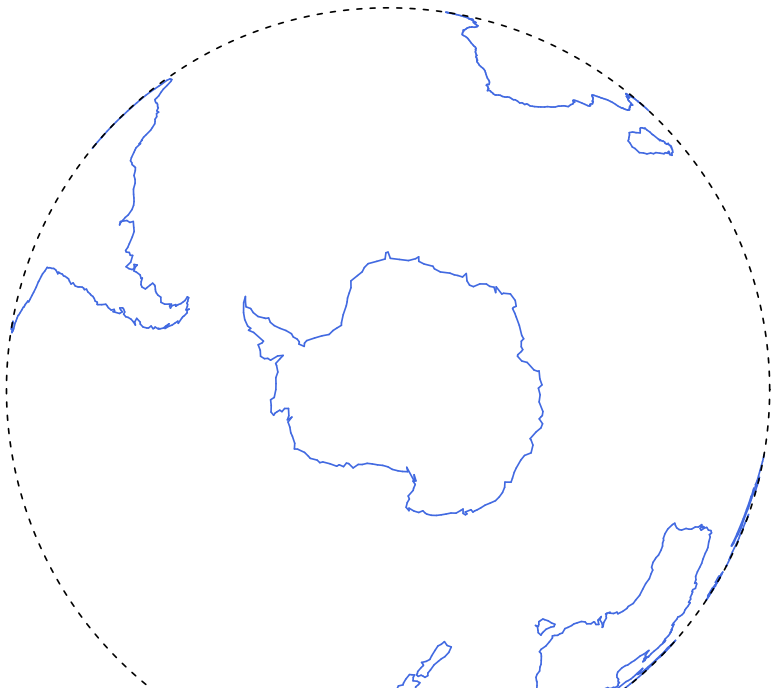
Coming back to the world


```
library(globe)
globeearth(eye=place("titanic"),col="royalblue")
```

Available places for `globeearth`

'nedlands', 'curtin', 'perth', 'northpole', 'southpole', 'casey', 'mawson', 'madrid', 'aarhus', 'aalborg', 'newyorkcity', 'titanic', 'pyongyang', 'everest', 'kilimanjaro'

```
globeearth(eye=place("southpole"),col="royalblue")
```



Resources

- github repo on osmplotr at **ropensci**; Intro to the package
- #30daymapchallenge
- Spatial Data Science

Further resources

- Tips for working with images in Rmd files