Spatial Data Science: using R as your command line GIS

Egge-Jan Pollé

FOSS4G Brussels, 25 October 2018

Two presentations about R

FOSS4G BELGIUM

ceci n'est pas une conférence

- 11:00 - 11:20:

  Spatial Data Science: Using R as your command line GIS,
  by Egge-Jan Pollé (Tensing)

- 11:25 - 11:45:

  Why and How to use R as an opensource GIS ? The Agromet project usecase, by Thomas Goossens (CRA-W)

R is FOSS4G

- R: a language and environment for statistical computing and graphics
- **FOSS:** yes, R is Free and Open Source Software
- **4G:** yes, thanks to additional libraries (packages) you can use it for Geospatial applications

In this presentation we will discuss three of those additional packages:

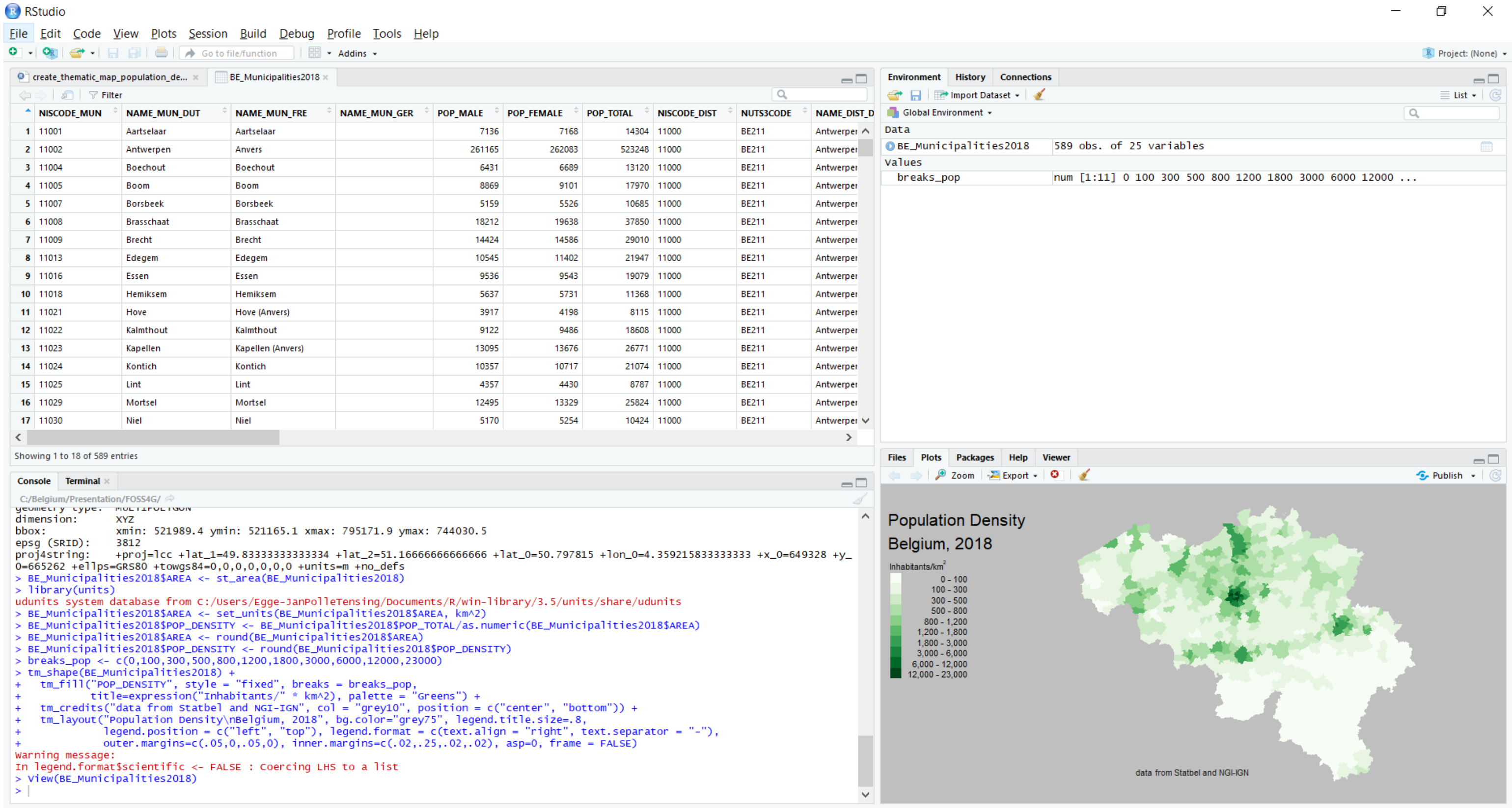
- **`sf`**
- **`tmap`**
- **`mapview`**

The R Eco system

R consortium

The Comprehensive R Archive Network: currently, the CRAN package repository features 13,236 available packages.

RStudio



	NISCODE_MUN	NAME_MUN_DUT	NAME_MUN_FRE	NAME_MUN_GER	POP_MALE	POP_FEMALE	POP_TOTAL	NISCODE_DIST	NUTS3CODE	NAME_DIST_D
1	11001	Aartselaar	Aartselaar		7136	7168	14304	11000	BE211	Antwerper
2	11002	Antwerpen	Anvers		261165	262083	523248	11000	BE211	Antwerper
3	11004	Boechout	Boechout		6431	6689	13120	11000	BE211	Antwerper
4	11005	Boom	Boom		8869	9101	17970	11000	BE211	Antwerper
5	11007	Borsbeek	Borsbeek		5159	5526	10685	11000	BE211	Antwerper
6	11008	Brasschaat	Brasschaat		18212	19638	37850	11000	BE211	Antwerper
7	11009	Brecht	Brecht		14424	14586	29010	11000	BE211	Antwerper
8	11013	Edegem	Edegem		10545	11402	21947	11000	BE211	Antwerper
9	11016	Essen	Essen		9536	9543	19079	11000	BE211	Antwerper
10	11018	Hemiksem	Hemiksem		5637	5731	11368	11000	BE211	Antwerper
11	11021	Hove	Hove (Anvers)		3917	4198	8115	11000	BE211	Antwerper
12	11022	Kalmthout	Kalmthout		9122	9486	18608	11000	BE211	Antwerper
13	11023	Kapellen	Kapellen (Anvers)		13095	13676	26771	11000	BE211	Antwerper
14	11024	Kontich	Kontich		10357	10717	21074	11000	BE211	Antwerper
15	11025	Lint	Lint		4357	4430	8787	11000	BE211	Antwerper
16	11029	Mortsel	Mortsel		12495	13329	25824	11000	BE211	Antwerper
17	11030	Niel	Niel		5170	5254	10424	11000	BE211	Antwerper

Showing 1 to 18 of 589 entries

```
C:/Belgium/Presentation/FOSS4G/
geometry type:  MULTIPOLYGON
dimension:      XYZ
bbox:           xmin: 521989.4 ymin: 521165.1 xmax: 795171.9 ymax: 744030.5
epsg (SRID):    3812
proj4string:    +proj=lcc +lat_1=49.83333333333334 +lat_2=51.16666666666666 +lat_0=50.797815 +lon_0=4.359215833333333 +x_0=649328 +y_
0=665262 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
> BE_Municipalities2018$AREA <- st_area(BE_Municipalities2018)
> library(units)
udunits system database from C:/Users/Egge-JanPolleTensing/Documents/R/win-library/3.5/units/share/udunits
> BE_Municipalities2018$AREA <- set_units(BE_Municipalities2018$AREA, km^2)
> BE_Municipalities2018$POP_DENSITY <- BE_Municipalities2018$POP_TOTAL/as.numeric(BE_Municipalities2018$AREA)
> BE_Municipalities2018$AREA <- round(BE_Municipalities2018$AREA)
> BE_Municipalities2018$POP_DENSITY <- round(BE_Municipalities2018$POP_DENSITY)
> breaks_pop <- c(0,100,300,500,800,1200,1800,3000,6000,12000,23000)
> tm_shape(BE_Municipalities2018) +
+   tm_fill("POP_DENSITY", style = "fixed", breaks = breaks_pop,
+           title=expression("Inhabitants/" * km^2), palette = "Greens") +
+   tm_credits("data from Statbel and NGI-IGN", col = "grey10", position = c("center", "bottom")) +
+   tm_layout("Population Density\nBelgium, 2018", bg.color="grey75", legend.title.size=.8,
+             legend.position = c("left", "top"), legend.format = c(text.align = "right", text.separator = "-"),
+             outer.margins=c(.05,0,.05,0), inner.margins=c(.02,.25,.02,.02), asp=0, frame = FALSE)
Warning message:
In legend.format$scientific <- FALSE : Coercing LHS to a list
> View(BE_Municipalities2018)
> 
```

```
Data
BE_Municipalities2018    589 obs. of 25 variables
Values
  breaks_pop             num [1:11] 0 100 300 500 800 1200 1800 3000 6000 12000 ...
```

RStudio (recommended): makes R easier to use. RStudio includes a code editor, debugging & visualization tools https://www.rstudio.com/

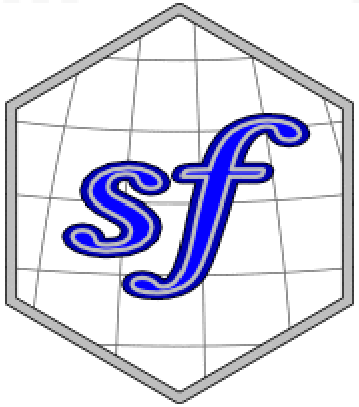
sf: Simple Features for R

This package provides support for simple features, which is a standardized way to encode spatial vector data

- https://www.r-consortium.org/blog/2017/01/03/simple-features-now-on-cran
- https://cran.r-project.org/package=sf

Presentations by Edzer Pebesma at useR!, July 2017, Brussels:

- Spatial data in R: new directions (video)
- Spatial data in R: new directions II (video)



The real geospatial powers behind `sf`

- GDAL: the **Geospatial Data Abstraction Library** is a translator library for raster and vector geospatial data formats.

  http://www.gdal.org/

- GEOS: the **Geometry Engine, Open Source** contains the complete functionality of the OpenGIS Simple Features for SQL spatial predicate functions and spatial operators.

  https://trac.osgeo.org/geos

- Proj.4: **PROJ** is a generic coordinate transformation software, that transforms coordinates from one coordinate reference system (CRS) to another. This includes cartographic projections as well as geodetic transformations.

  http://proj4.org/

You can see this when you load **`sf`** into R:

```
> library(sf)
Linking to GEOS 3.6.1, GDAL 2.2.3, proj.4 4.9.3
```

Convert a data.frame to an sf object

```
> BE_Airports_csv
                              airport iata latitude longitude
1       Antwerp International Airport  ANR 51.18944  4.460278
2                    Brussels Airport  BRU 50.90139  4.484444
3                       Liege Airport  LGG 50.63639  5.442778
4    Brussels South Charleroi Airport  CRL 50.46000  4.452778
5 Ostend-Bruges International Airport  OST 51.19889  2.862222
> class(BE_Airports_csv)
[1] "data.frame"
> BE_Airports <- st_as_sf(BE_Airports_csv,
                       coords = c('longitude','latitude'), crs = 4326)
> class(BE_Airports)
[1] "sf"         "data.frame"
```

Simple Feature - with geometry column!

```
> BE_Airports
Simple feature collection with 5 features and 2 fields
geometry type:  POINT
dimension:      XY
bbox:           xmin: 2.862222 ymin: 50.46 xmax: 5.442778 ymax: 51.19889
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
                            airport iata                   geometry
1       Antwerp International Airport  ANR POINT (4.460278 51.18944)
2                    Brussels Airport  BRU POINT (4.484444 50.90139)
3                       Liege Airport  LGG POINT (5.442778 50.63639)
4    Brussels South Charleroi Airport  CRL    POINT (4.452778 50.46)
5 Ostend-Bruges International Airport  OST POINT (2.862222 51.19889)
```

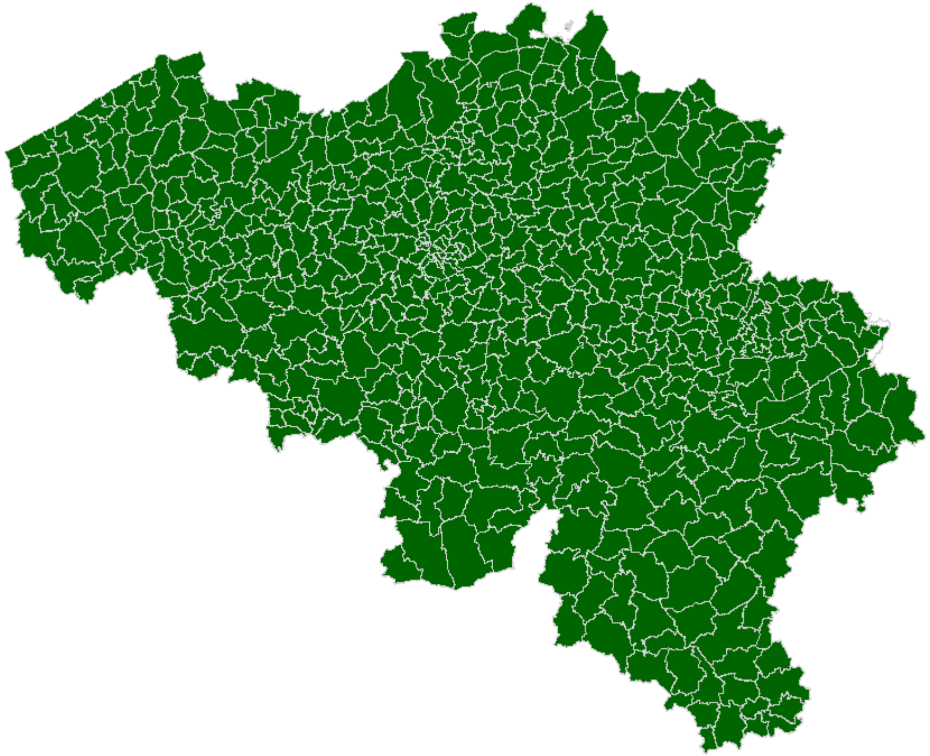
Read spatial data

For this presentation a dataset has been compiled based on data from Statbel and NGI-IGN.
You can download the file Belgium2018.json using this piece of R code.

```
> BE_Municipalities2018 <- st_read("./Data/Belgium2018.json")
Reading layer `Belgium2018' from data source
`C:\Belgium\Presentation\FOSS4G\Data\Belgium2018.json' using driver `GeoJSON'
Simple feature collection with 589 features and 22 fields
geometry type:  MULTIPOLYGON
dimension:      XYZ
bbox:           xmin: 521989.4 ymin: 521165.1 xmax: 795171.9 ymax: 744030.5
epsg (SRID):    3812
proj4string:    +proj=lcc +lat_1=49.83333333333334 +lat_2=51.16666666666666
+lat_0=50.797815 +lon_0=4.359215833333333 +x_0=649328 +y_0=665262
+ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
```

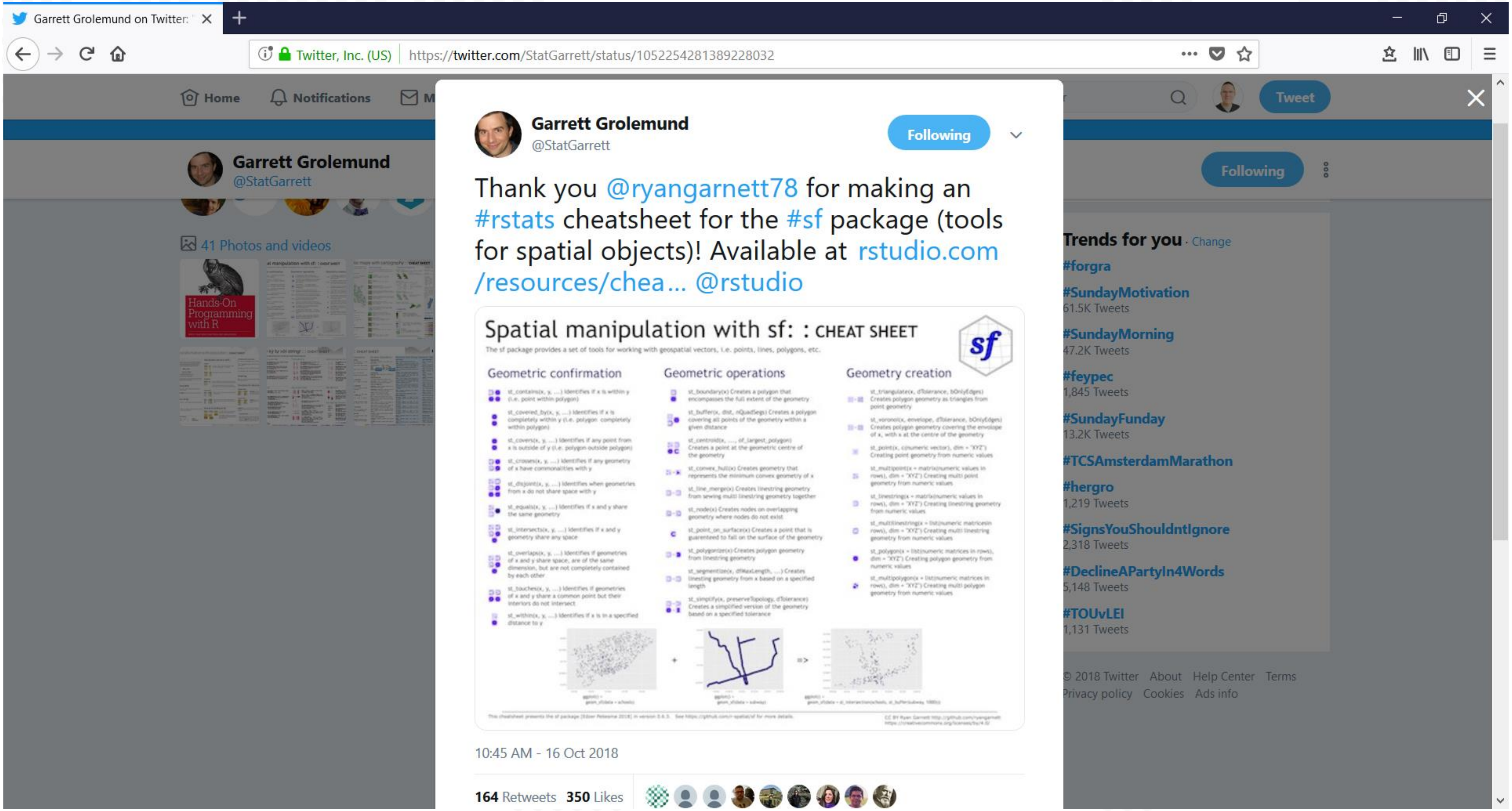
Plot BE_Municipalities2018

```
> plot(st_geometry(BE_Municipalities2018), col = "darkgreen", border =
"lightgray" )
```



Methods for sf objects

```
> methods(class = "sf")
 [1] $<-                [                     [[<-               aggregate             as.data.frame
 [6] cbind              coerce                dbDataType         dbWriteTable          identify
[11] initialize         mapView               merge              plot                  print
[16] rbind              show                  slotsFromS3        st_agr                st_agr<-
[21] st_as_sf           st_bbox               st_boundary        st_buffer             st_cast
[26] st_centroid        st_collection_extract st_convex_hull     st_coordinates        st_crs
[31] st_crs<-           st_difference         st_geometry        st_geometry<-         st_intersection
[36] st_is              st_line_merge         st_node            st_point_on_surface   st_polygonize
[41] st_precision       st_segmentize         st_set_precision   st_simplify           st_snap
[46] st_sym_difference  st_transform          st_triangulate     st_union              st_voronoi
[51] st_wrap_dateline   st_write              st_zm
```



Garrett Grolemund on Twitter:
https://twitter.com/StatGarrett/status/1052254281389228032
Twitter, Inc. (US)
Home
Notifications
Tweet
Garrett Grolemund
@StatGarrett
Following
41 Photos and videos
Hands-On Programming with R

Garrett Grolemund
@StatGarrett

Following

Thank you @ryangarnett78 for making an #rstats cheatsheet for the #sf package (tools for spatial objects)! Available at rstudio.com/resources/chea… @rstudio

Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.

Geometric confirmation

- st_contains(x, y, ...) Identifies if x is within y (i.e. point within polygon)
- st_covered_by(x, y, ...) Identifies if x is completely within y (i.e. polygon completely within polygon)
- st_covers(x, y, ...) Identifies if any point from x is outside of y (i.e. polygon outside polygon)
- st_crosses(x, y, ...) Identifies if any geometry of x have commonalities with y
- st_disjoint(x, y, ...) Identifies when geometries from x do not share space with y
- st_equals(x, y, ...) Identifies if x and y share the same geometry
- st_intersects(x, y, ...) Identifies if x and y geometry share any space
- st_overlaps(x, y, ...) Identifies if geometries of x and y share space, are of the same dimension, but are not completely contained by each other
- st_touches(x, y, ...) Identifies if geometries of x and y share a common point but their interiors do not intersect
- st_within(x, y, ...) Identifies if x is in a specified distance to y

Geometric operations

- st_boundary(x) Creates a polygon that encompasses the full extent of the geometry
- st_buffer(x, dist, nQuadSegs) Creates a polygon covering all points of the geometry within a given distance
- st_centroid(x, ..., of_largest_polygon) Creates a point at the geometric centre of the geometry
- st_convex_hull(x) Creates geometry that represents the minimum convex geometry of x
- st_line_merge(x) Creates linestring geometry from sewing multi linestring geometry together
- st_node(x) Creates nodes on overlapping geometry where nodes do not exist
- st_point_on_surface(x) Creates a point that is guarenteed to fall on the surface of the geometry
- st_polygonize(x) Creates polygon geometry from linestring geometry
- st_segmentize(x, dfMaxLength, ...) Creates linesting geometry from x based on a specified length
- st_simplify(x, preserveTopology, dTolerance) Creates a simplified version of the geometry based on a specified tolerance

Geometry creation

- st_triangulate(x, dTolerance, bOnlyEdges) Creates polygon geometry as triangles from point geometry
- st_voronoi(x, envelope, dTolerance, bOnlyEdges) Creates polygon geometry covering the envelope of x, with x at the centre of the geometry
- st_point(x, c(numeric vector), dim = "XYZ") Creating point geometry from numeric values
- st_multipoint(x = matrix(numeric values in rows), dim = "XYZ") Creating multi point geometry from numeric values
- st_linestring(x = matrix(numeric values in rows), dim = "XYZ") Creating linestring geometry from numeric values
- st_multilinestring(x = list(numeric matricesin rows), dim = "XYZ") Creating multi linestring geometry from numeric values
- st_polygon(x = list(numeric matrices in rows), dim = "XYZ") Creating polygon geometry from numeric values
- st_multipolygon(x = list(numeric matrices in rows), dim = "XYZ") Creating multi polygon geometry from numeric values

10:45 AM - 16 Oct 2018

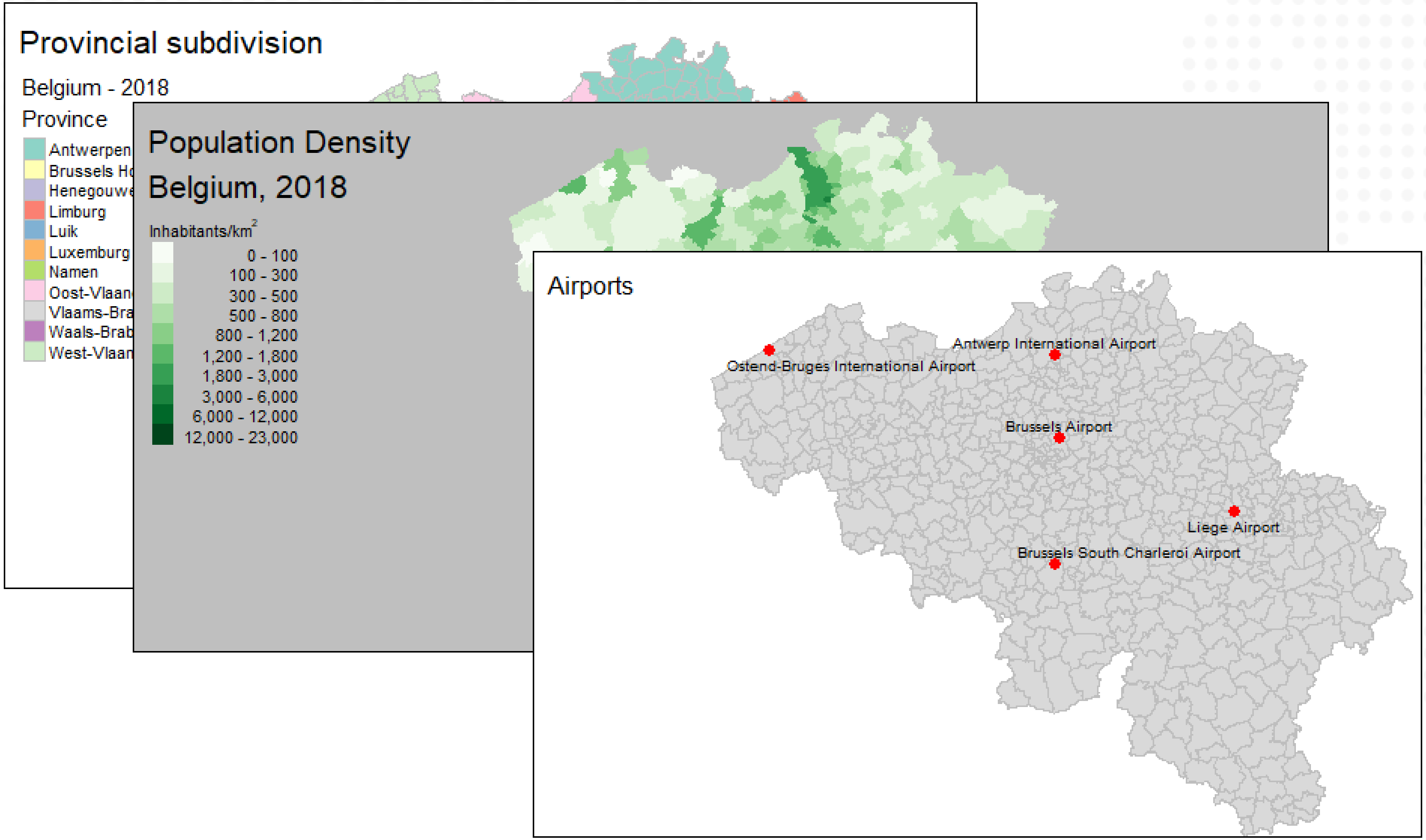
164 Retweets 350 Likes

Trends for you · Change

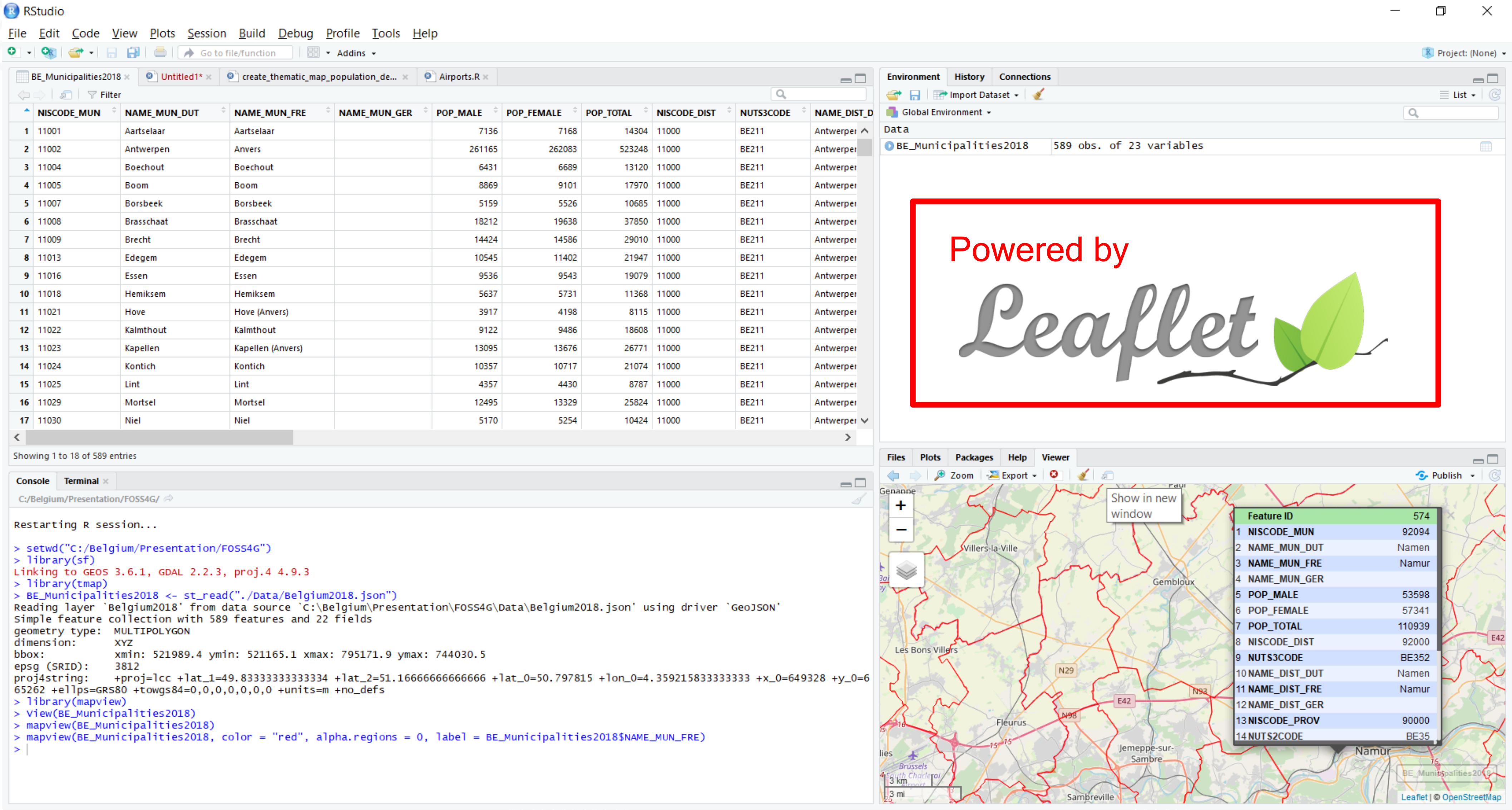
- #forgra
- #SundayMotivation 61.5K Tweets
- #SundayMorning 47.2K Tweets
- #feypec 1,845 Tweets
- #SundayFunday 13.2K Tweets
- #TCSAmsterdamMarathon
- #hergro 1,219 Tweets
- #SignsYouShouldntIgnore 2,318 Tweets
- #DeclineAPartyIn4Words 5,148 Tweets
- #TOUvLEI 1,131 Tweets

© 2018 Twitter About Help Center Terms Privacy policy Cookies Ads info

Thematic mapping in R with the package `tmap`



Interactive mapping in R with the package **`mapview`**



	NISCODE_MUN	NAME_MUN_DUT	NAME_MUN_FRE	NAME_MUN_GER	POP_MALE	POP_FEMALE	POP_TOTAL	NISCODE_DIST	NUTS3CODE	NAME_DIST_D
1	11001	Aartselaar	Aartselaar		7136	7168	14304	11000	BE211	Antwerper
2	11002	Antwerpen	Anvers		261165	262083	523248	11000	BE211	Antwerper
3	11004	Boechout	Boechout		6431	6689	13120	11000	BE211	Antwerper
4	11005	Boom	Boom		8869	9101	17970	11000	BE211	Antwerper
5	11007	Borsbeek	Borsbeek		5159	5526	10685	11000	BE211	Antwerper
6	11008	Brasschaat	Brasschaat		18212	19638	37850	11000	BE211	Antwerper
7	11009	Brecht	Brecht		14424	14586	29010	11000	BE211	Antwerper
8	11013	Edegem	Edegem		10545	11402	21947	11000	BE211	Antwerper
9	11016	Essen	Essen		9536	9543	19079	11000	BE211	Antwerper
10	11018	Hemiksem	Hemiksem		5637	5731	11368	11000	BE211	Antwerper
11	11021	Hove	Hove (Anvers)		3917	4198	8115	11000	BE211	Antwerper
12	11022	Kalmthout	Kalmthout		9122	9486	18608	11000	BE211	Antwerper
13	11023	Kapellen	Kapellen (Anvers)		13095	13676	26771	11000	BE211	Antwerper
14	11024	Kontich	Kontich		10357	10717	21074	11000	BE211	Antwerper
15	11025	Lint	Lint		4357	4430	8787	11000	BE211	Antwerper
16	11029	Mortsel	Mortsel		12495	13329	25824	11000	BE211	Antwerper
17	11030	Niel	Niel		5170	5254	10424	11000	BE211	Antwerper

Showing 1 to 18 of 589 entries

Data

BE_Municipalities2018 589 obs. of 23 variables

Powered by Leaflet

Console Terminal

C:/Belgium/Presentation/FOSS4G/

```
Restarting R session...

> setwd("C:/Belgium/Presentation/FOSS4G")
> library(sf)
Linking to GEOS 3.6.1, GDAL 2.2.3, proj.4 4.9.3
> library(tmap)
> BE_Municipalities2018 <- st_read("./Data/Belgium2018.json")
Reading layer `Belgium2018' from data source `C:\Belgium\Presentation\FOSS4G\Data\Belgium2018.json' using driver `GeoJSON'
Simple feature collection with 589 features and 22 fields
geometry type:  MULTIPOLYGON
dimension:      XYZ
bbox:           xmin: 521989.4 ymin: 521165.1 xmax: 795171.9 ymax: 744030.5
epsg (SRID):    3812
proj4string:    +proj=lcc +lat_1=49.83333333333334 +lat_2=51.16666666666666 +lat_0=50.797815 +lon_0=4.359215833333333 +x_0=649328 +y_0=6
65262 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
> library(mapview)
> View(BE_Municipalities2018)
> mapview(BE_Municipalities2018)
> mapview(BE_Municipalities2018, color = "red", alpha.regions = 0, label = BE_Municipalities2018$NAME_MUN_FRE)
> 
```

	Feature ID	574
1	NISCODE_MUN	92094
2	NAME_MUN_DUT	Namen
3	NAME_MUN_FRE	Namur
4	NAME_MUN_GER	
5	POP_MALE	53598
6	POP_FEMALE	57341
7	POP_TOTAL	110939
8	NISCODE_DIST	92000
9	NUTS3CODE	BE352
10	NAME_DIST_DUT	Namen
11	NAME_DIST_FRE	Namur
12	NAME_DIST_GER	
13	NISCODE_PROV	90000
14	NUTS2CODE	BE35

A short note on the package sp

sp: Classes and Methods for Spatial Data

https://cran.r-project.org/package=sp

- sp is the predecessor of sf
- So, sf is the successor of sp :-)
- sp has been developed by the s ors as sf
- You will definitely encounter s ou google for R and Spatial… but our recommend stick with sf, and try to forget about sp

Pebesma: “The package sf aims at succeeding sp in the long term.”

Egge-Jan Pollé

ejpolle@tensing.com
https://www.linkedin.com/in/ejhpolle
@EggePolle

Thank you for your attention

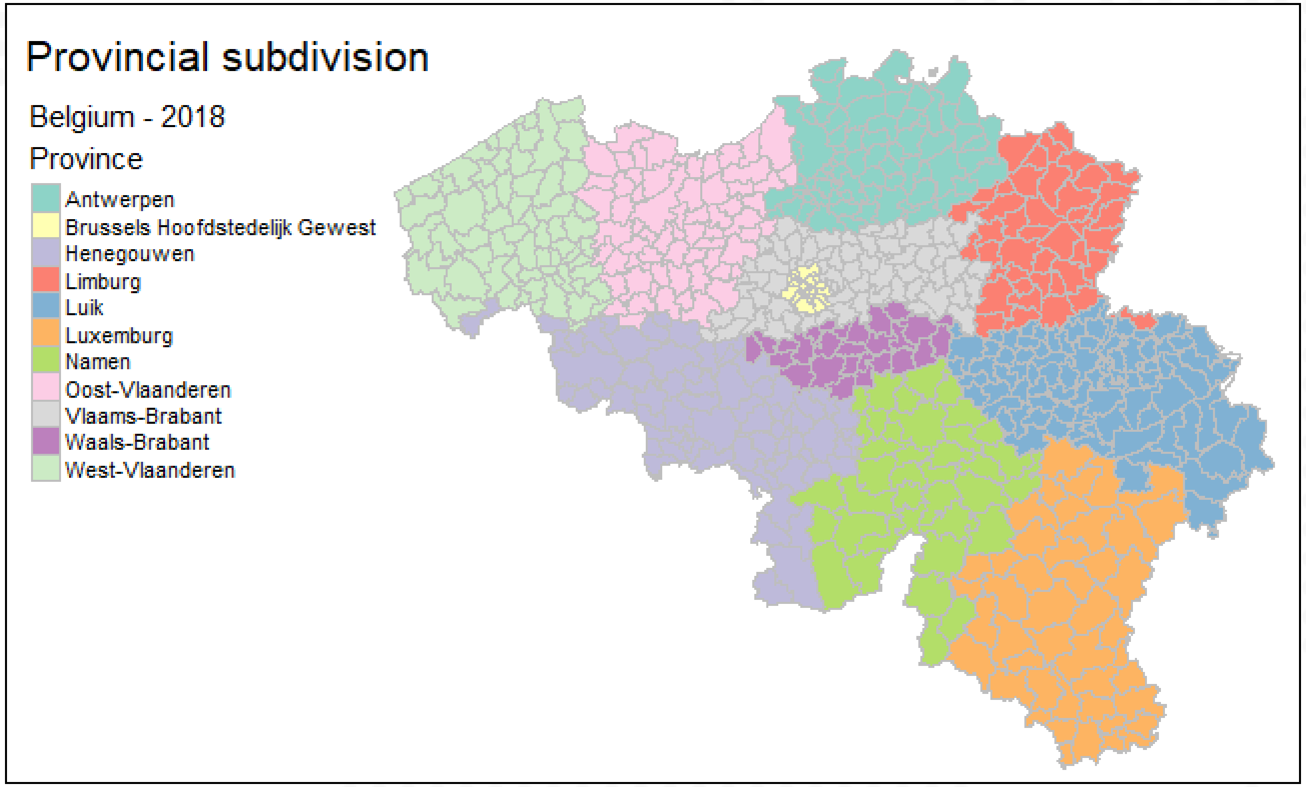
Appendix Download Sample Data **`Belgium2018.json`**

R code:

```
# Store the URL to the file to download in a variable

URL2zip <- "http://www.twiav.nl/files/Belgium2018.zip"

# Create a temporary file

zip_file <- tempfile(fileext = ".zip")

# Download the file

download.file(URL2zip, destfile = zip_file, mode = "wb")

# Create a subfolder in your working directory to store the unzipped data

dir.create("./Data", showWarnings = FALSE)

# Unzip the file

unzip(zip_file, exdir = "./Data")

# After unzipping you can delete (i.e. unlink) the file

unlink(zip_file)

# Remove variables you do not longer need

rm(URL2zip, zip_file)
```

Now you are ready to load the data into R using the function st_read()

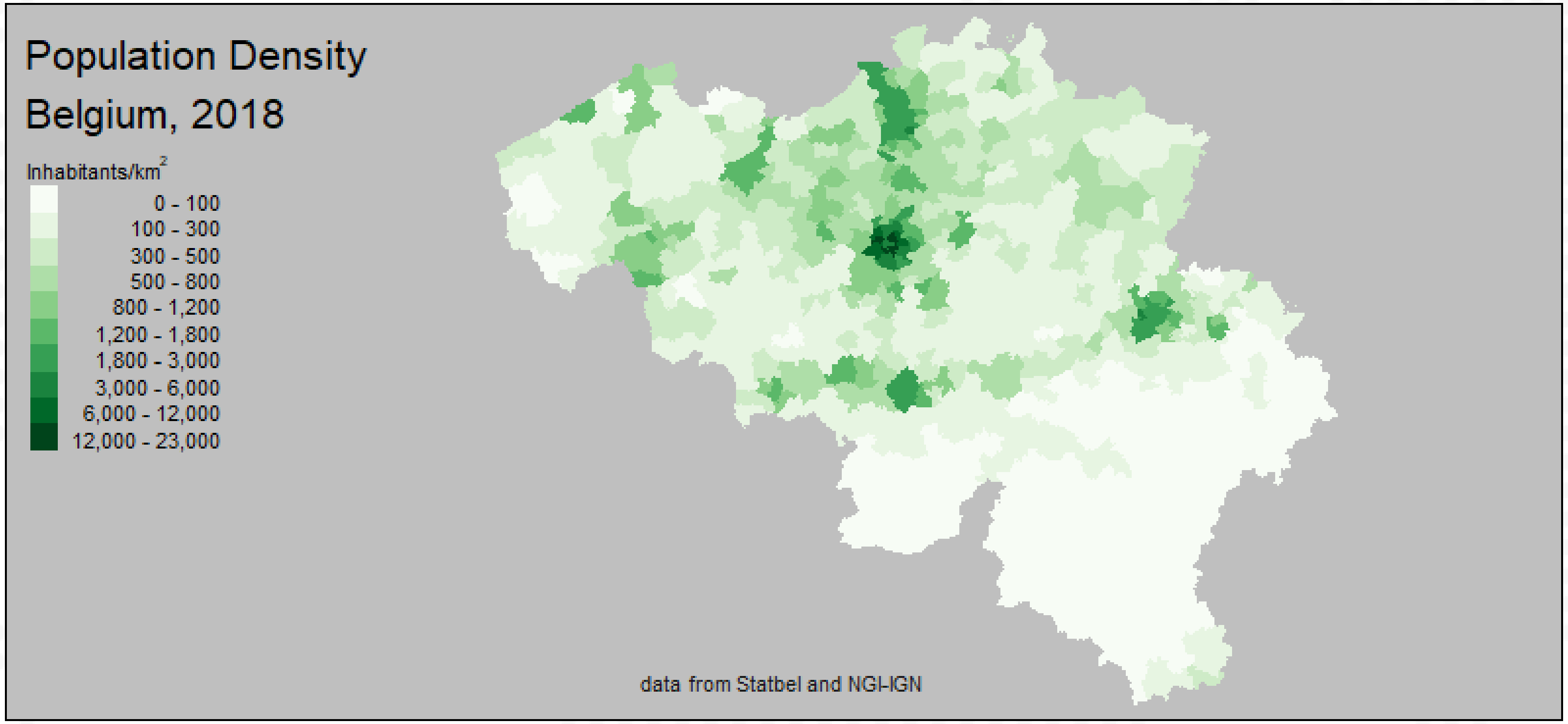
Appendix tamp - 1



R code:

```
qtm(shp = BE_Municipalities2018,
    title = "Provincial subdivision",
    fill = "NAME_PROV_DUT",
    fill.title = "Belgium - 2018\nProvince",
    borders = "grey",
    format = "NLD_wide",
    frame = TRUE)
```

Appendix **tamp** - 2



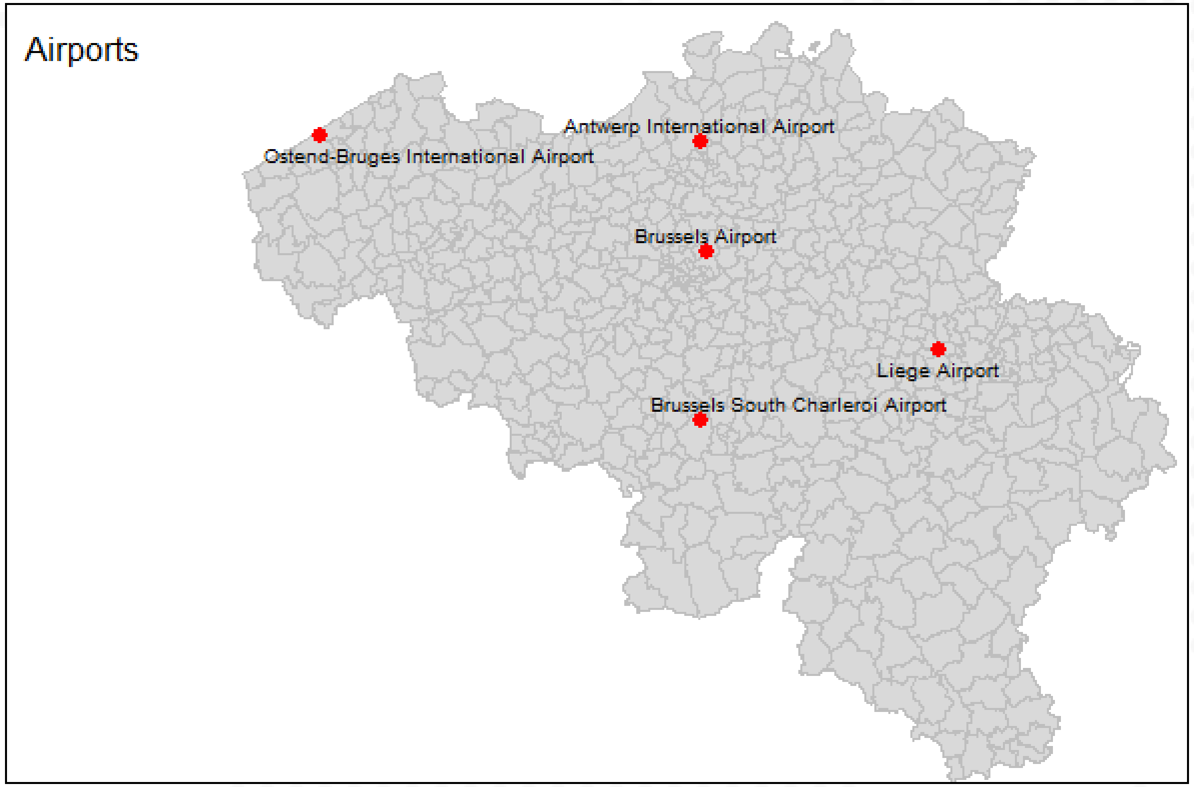
R code:

```
tm_shape(BE_Municipalities2018) +
  tm_fill("POP_DENSITY", style = "fixed", breaks = breaks_pop,
          title=expression("Inhabitants/" * km^2), palette = "Greens") +
  tm_credits("data from Statbel and NGI-IGN", col = "grey10", position = c("center", "bottom")) +
  tm_layout("Population Density\nBelgium, 2018", bg.color="grey75", legend.title.size=.8,
            legend.position = c("left", "top"), legend.format = c(text.align = "right",
            text.separator = "-"), outer.margins=c(.05,0,.05,0),
            inner.margins=c(.02,.25,.02,.02), asp=0, frame = TRUE)
```

Appendix **tamp** - 3

R code:

```
tm_shape(BE_Municipalities2018) +
  tm_polygons(border.col = "grey")+
  tm_shape(BE_Airports) +
  tm_dots(size = .5, col = "red",
          palette = "Set1", popup.vars = TRUE) +
  tm_text("airport", size =.8,
          legend.size.show = FALSE,
          root=8, size.lowerbound = .7,
          auto.placement = TRUE) +
  tm_style("white", title = "Airports") +
  tm_format("World_wide")
```


Airports
Antwerp International Airport
Ostend-Bruges International Airport
Brussels Airport
Liege Airport
Brussels South Charleroi Airport