Spatial Microsimulation: an Example with German Data

Author: Esteban Munoz emunozh@gmail.com
Date: Friday 5th December, 2014
Prepared for: The AURIN/NATSEM Microsimulation Symposium

Table of Contents

What you need to run this simulation 1
Getting the data and scripts to run the simulation 2
Data Stucture of the Census 2011 and the Micro census 2002 3
- Age 3
- Marital status 4
- Household size 4

Running the Simulation 5
- Load the micro census survey data 5
- Rearrange the data to fit the Census data 6
- Load the census data 6
- Select the state to simulate 7
- Remove unwanted columns from the data 8
- Prepare data for simulation 8
- Get the new weights for each area 9
- Save the result to a csv file 9
- Make some nice plots with the result 9

Visualizing the results in QGIS 11
Simulation Results 14
Annex: Scripts to fetch the required data from the internet 16

What you need to run this simulation

In order to run this simulation you need:

1. A working version of R. This simulation was tested with version 3.1.1 (2014-07-10) on a 64-bit linux-gnu machine.
2. The simulation requires 2 additional R packages: MASS and GREGWT. The first library is available at the cran repositories [1]:

   You can install the MASS library with this command:

   ```
   install.packages("MASS", dependencies = TRUE)
   ```

   In this simulation I use version 7.3-33 of the MASS library.

The second library is currently under development and can't be downloaded from the cran repository. In order to get this library you will have to install the library 'from source'. You will need to download the source code from the provided link (see data below) and type the following command on the R command line:

```
install.packages("C:\\GREGWT_1.4.tar.gz", repos=NULL, type="source")
```

3. Two datasets: a) the census data from the last available census (2011) for the desire simulation region; and b) the micro census survey, this survey is only available, without any restriction, for the year 2002.

   The census data can be downloaded from the official census website [2] maintain by the Federal Statistics Bureau of Germany (Statistische Ämter des Bundes und der Länder). Nonetheless, the data has to be downloaded manually where the input of every single area has to be explicitly given as a variable. I wrote a small script that automatize this process (see Annex: Scripts to fetch the required data from the internet). You can download the extracted data from the links provided below.

   The micro census survey can be downloaded from the research data center [3] also maintain by the Federal Statistics Bureau of Germany. This data set can be downloaded in different file formats, we will use the **.csv* file format.

4. The Federal Statistics Bureau also provides a shape file **.shp* for the visualization of data.
5. A GIS system that can open shape files and merge a csv file to the file. In this example I use QGIS [4], this is an open source, platform independent GIS system. In this example I use QGIS version 2.6.

Getting the data and scripts to run the simulation

In order to facilitate the work flow of this example I

Assumed folder structure of the downloaded zip files:

```
./--> mikrosim.R [Required]
./--> MicrosimulationGerman.pdf [This file]

./Data/ [Required]
    |
    +--> Gemeinden/
    |       |
    |       +--> ALTER_AF-all.csv
    |       +--> FAMSTND_KURZ-all.csv
    |       +--> HHGROESS_KLASS-all.csv
    |
    +--> Survey/
    |       |
    |       +--> mz02_cf.csv
    |
    +--> Shapefiles/
            |
            +--> VG250_Gemeinden.shp .dfb .prj .shx

./Steps/ [Optional]
    |
    +--> 01.RData
    +--> 02.RData
    +--> 03.RData
    +--> de.RData
    +--> SimulationResult.csv
```

```
    +--> SimulationResult-de.csv

./Doc/
    |
    +--> Datensatzbeschreibung.pdf
    +--> fdz_mikrozensus_cf_2002_schluesselverzeichnis.pdf
    +--> HeatExpenditure.jpeg
    +--> HeatExpenditureHist.jpeg
    +--> map.jpeg
    +--> map-de.jpeg
    +--> MicrosimulationGerman.rst
    +--> Screenshot1.png
    +--> Screenshot1-1.png
    +--> Screenshot2.png
    +--> Screenshot2-1.png
    +--> Screenshot3.png
    +--> Screenshot3-1.png
    +--> Screenshot4.png
    +--> Screenshot5.png
    +--> Screenshot6.png

./Extra/
    |
    +--> AGS.csv
    +--> AGS-Gemeinden.csv
    +--> cleanAGS.py
    +--> getData.sh
```

Data Stucture of the Census 2011 and the Micro census 2002

Age

Census 2011	**Micro census 2002**
Under.3	[0 ... 94]
X3...5	
X6...14	
X15...17	
X18...24	
X25...29	
X30...39	
X40...49	
X50...64	
X65...74	
X75.and.over	
	95 (>= 95)

Marital status

Census 2011	Micro census 2002
Single	1 (Ledig -- Single)
Married ...	2 (Verheiratet -- Married)
Widowed ...	3 (Verwitwet -- Widowed)
Divorced ...	4 (Geschieden -- Divorced)
No.Data	

Household size

Census 2011	Micro census 2002
1.person	[1...8]
2.persons	
3.persons	
4.persons	
5.persons	
6.or.more.people	
	>= 9
	0 (Other)

Running the Simulation

Load the micro census survey data

```
mikro.raw = read.csv("./DATA/Survey/mz02_cf.csv", sep=";")
# columns to keep for simulation:
# age, marital status, household size, weights
keep.simulation = c(
    "ef30",   # Age
    "ef35",   # Marital status
    "ef521",  # Household size
    "ef750")  # Weights
mikro.simulation <- mikro.raw[names(mikro.raw) %in% keep.simulation]
# columns to keep for result:
# cold operating cost, warm operating cost
keep.result = c("ef464", "ef466")
mikro.result <- mikro.raw[names(mikro.raw) %in% keep.result]
mikro.result$ef464[mikro.result$ef464 == 9998]  <- 450
mikro.result$ef464[mikro.result$ef464 == 9999]  <- NA
mikro.result$ef464[mikro.result$ef464 == 8]     <- NA
mikro.result$ef464[mikro.result$ef466 == 9998]  <- 450
mikro.result$ef464[mikro.result$ef466 == 9999]  <- NA
mikro.result$ef464[mikro.result$ef466 == 8]     <- NA
mikro.result <- mikro.result$ef466 - mikro.result$ef464
# remove all observations with NaN values
mikro.simulation <- mikro.simulation[complete.cases(mikro.result),]
mikro.result <- mikro.result[complete.cases(mikro.result)]
```

Rearrange the data to fit the Census data

```
# Age
# Create empy vectors
age.01 <- vector(length=dim(mikro.simulation)[1])
age.02 <- vector(length=dim(mikro.simulation)[1])
...
age.11 <- vector(length=dim(mikro.simulation)[1])

# Fill the vectors with boolean values
age.01[mikro.simulation$ef30 < 3] = 1
age.02[mikro.simulation$ef30 < 6 & mikro.simulation$ef30 >= 3] = 1
...
age.11[mikro.simulation$ef30 >= 75] = 1

# Marital status
mst.01 <- vector(length=dim(mikro.simulation)[1])
...
mst.01[mikro.simulation$ef35 == 1] = 1
...

# Household size
hhs.01 <- vector(length=dim(mikro.simulation)[1])
...
hhs.01[mikro.simulation$ef521 == 1] = 1
...

# Put everything on a data frame
mikro.input = data.frame(
    age.01 = age.01,
    ...
    mst.01 = mst.01,
    ...
    hhs.01 = hhs.01,
    ...
    hhs.06 = hhs.06)

# And the vector with the weights
dx <- mikro.simulation$ef750
```

Load the census data

```
nan.strings = c('nan', '.')
gem.alt = read.csv("./DATA/Gemeinden/ALTER_AF-all.csv",
    colClasses=c("character",rep("numeric",6)),
    na.strings = nan.strings)
gem.fam = read.csv("./DATA/Gemeinden/FAMSTND_KURZ-all.csv",
    colClasses=c("character",rep("numeric",6)),
    na.strings = nan.strings)
gem.hhs = read.csv("./DATA/Gemeinden/HHGROESS_KLASS-all.csv",
    colClasses=c("character",rep("numeric",7)),
    na.strings = nan.strings)
```

Select the state to simulate

The following code simply filters the areas from the census tables given the first n letters of an area code. In this case the first two letters represent the code for the German federal states (see table state below for a complete list of the state codes).

```
# Select a single federal state (eg: 05 is the code for Nordrhein-Westfalen)
AGS.code = "05"
AGS.length = 2
gem.alt <- gem.alt[substr(gem.alt$X,1,AGS.length)==AGS.code, ]
gem.fam <- gem.fam[substr(gem.fam$X,1,AGS.length)==AGS.code, ]
gem.hhs <- gem.hhs[substr(gem.hhs$X,1,AGS.length)==AGS.code, ]
```

State	Code
Schleswig-Holstein	01
Hamburg	02
Niedersachsen	03
Bremen	04
Nordrhein-Westfalen	05
Hessen	06
Rheinland-Pfalz	07
Baden-Württemberg	08
Bayern	09
Saarland	10
Berlin	11
Brandenburg	12
Mecklenburg-Vorpommern	13
Sachsen	14
Sachsen-Anhalt	15
Thüringen	16

Remove unwanted columns from the data

```
# but save the total population first
population <- gem.alt$Total
# age
drop <- c("Total")
gem.alt <- gem.alt[,!(names(gem.alt) %in% drop)]
# marital status
drop <- c("Total", "No.data")
gem.fam <- gem.fam[,!(names(gem.fam) %in% drop)]
# Household size
drop <- c("Total")
gem.hhs <- gem.hhs[,!(names(gem.hhs) %in% drop)]

# Merge all data into a big data frame
gem.input <- merge(gem.alt, gem.fam, by.x = "X", by.y = "X")
gem.input <- merge(gem.input, gem.hhs, by.x = "X", by.y = "X")

# define the number of areas to run
#areas.number = 4
areas.number = dim(gem.input)[1]

# create a data frame to store the result
Result = data.frame(
    area=vector(length=areas.number),
    heat=vector(length=areas.number))
```

Prepare data for simulation

```
area.code <- gem.input[, 1]
Tx.s <- gem.input[, 2:dim(gem.input)[2]]
Simulation.Data <- prepareData(mikro.input, Tx.s)
mikro.input.s <- Simulation.Data$X
Tx.s <- Simulation.Data$Tx
X.in <- Simulation.Data$X.in
```

Get the new weights for each area

```
# loop through all areas
for(i in seq(1, areas.number)){

    # Create a vector with the area totals
    Tx <- Tx.s[i,]
    names(Tx) <- names(mikro.input.s)

    # Store the area code
    acode <- area.code[i]

    # Get new weights with GREGWT
    Weights = GREGWT(mikro.input.s, dx, Tx, bounds=c(0,Inf), X.input=X.in)
    fw <- Weights$Final.Weights
    # Compute average heat expenditure for this area
    heat.expenditure <- sum(mikro.result * fw / sum(fw), na.rm=TRUE)
    Result[i,] <- c(acode, heat.expenditure)}
```

Save the result to a csv file

```
Result <- Result[Result$heat > 0, ]
write.csv(Result, file="SimulationResult.csv")
```

Make some nice plots with the result

```
heat <- as.numeric(Result$heat)
jpeg(filename="HeatExpenditure.jpeg", width=600, height=600)
plot(sort(heat),
    main="Heat expenditure in German municipalities",
    ylab="Monthly heat expediture in EUR",
    xlab="Sorted municipalities")
abline(h=mean(heat, na.rm=TRUE), col='red', lw=3)
dev.off()
```

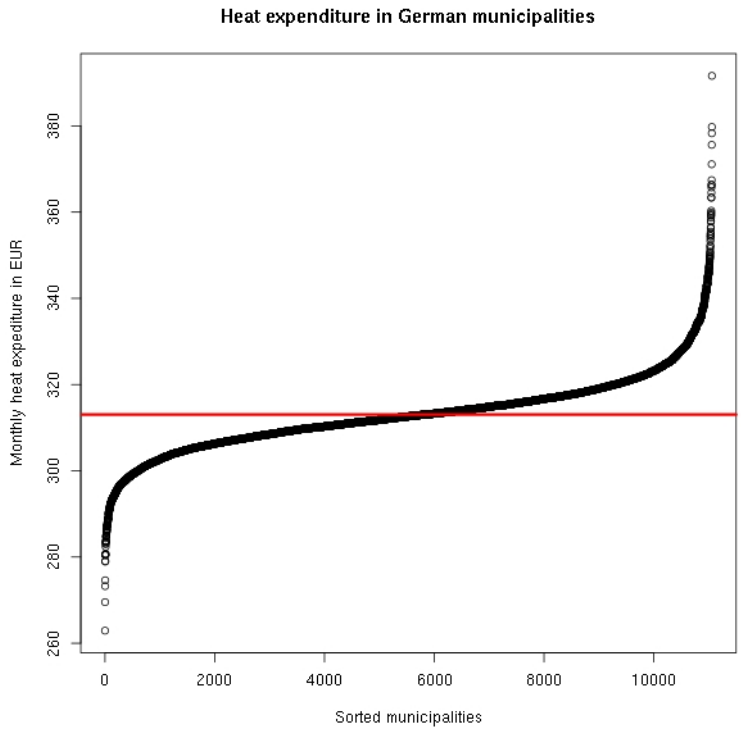

Figure : *Sorted heat expenditures of German municipalities*

```
1 jpeg(filename="HeatExpenditureHist.jpeg", width=600, height=600)
2 hist(heat, main="Histogram of heat expediture in German municipalities")
3 dev.off()
```

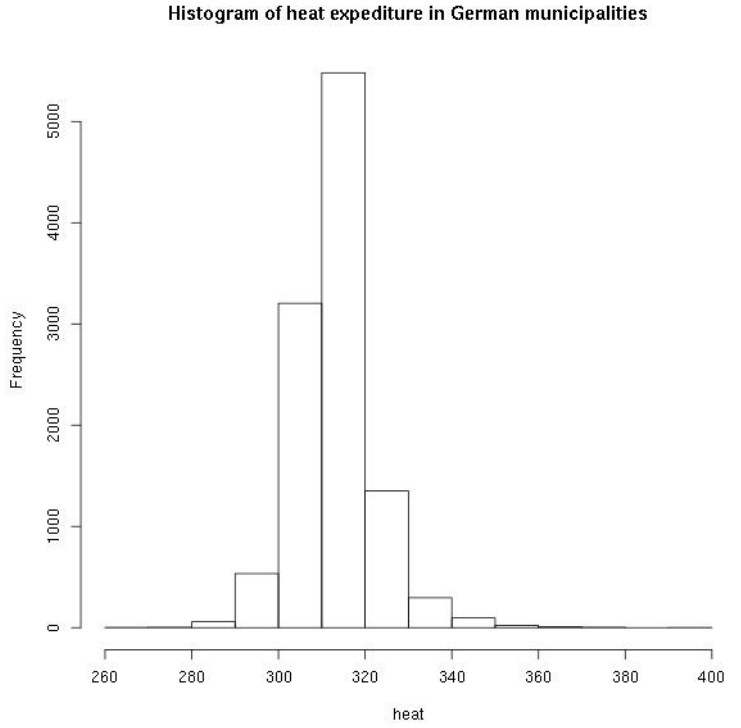

Figure : *Histogram of the heat expenditure for the German municipalities*

Visualizing the results in QGIS

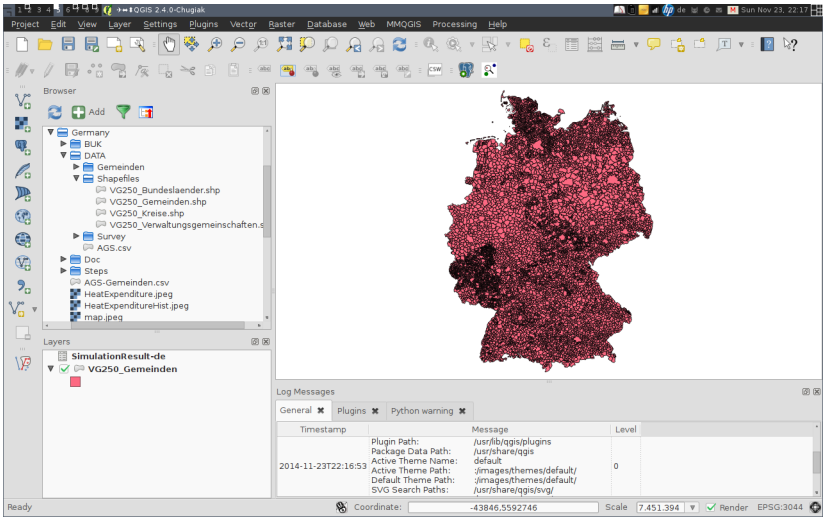

Figure : *Screenshot*

Figure : *Screenshot*

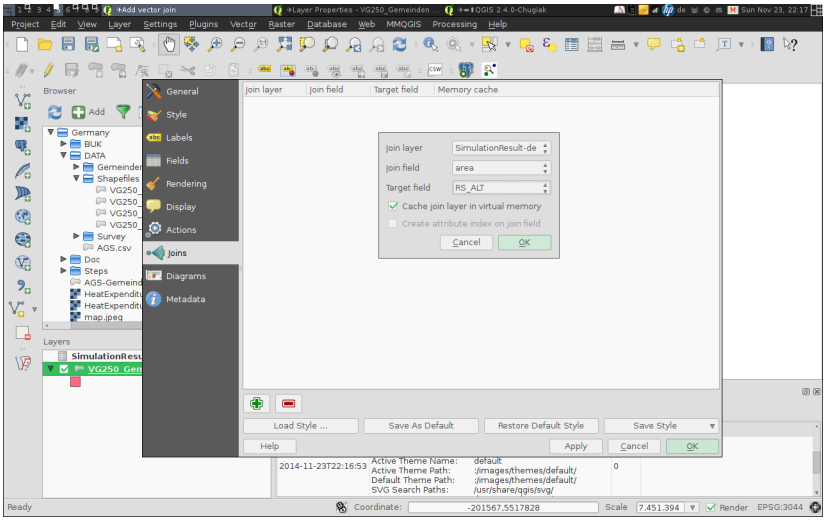

Figure : *Screenshot*

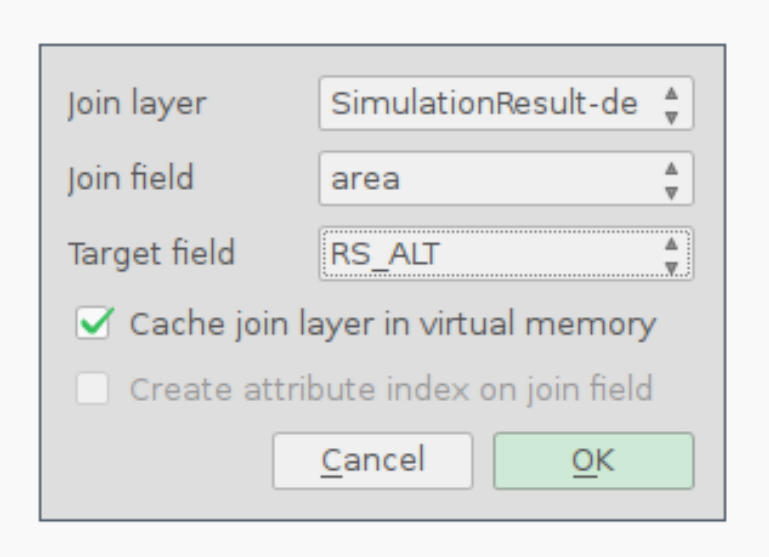

Figure : *Screenshot*

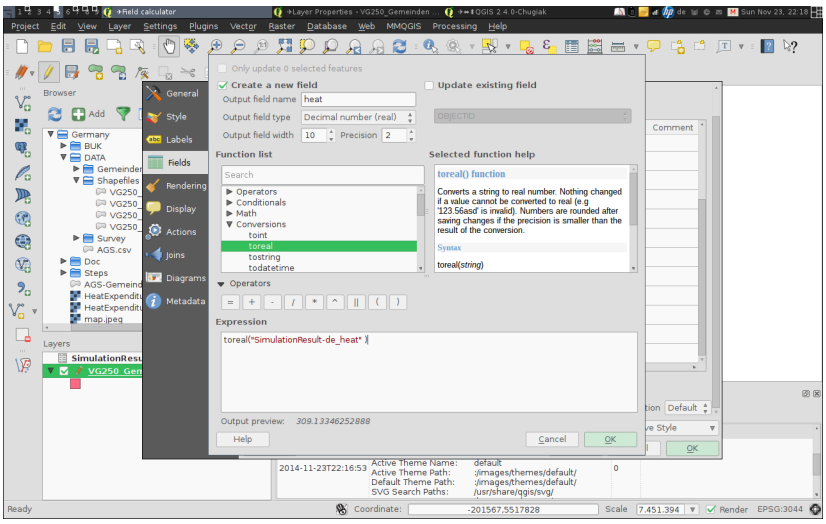

Figure : *Screenshot*

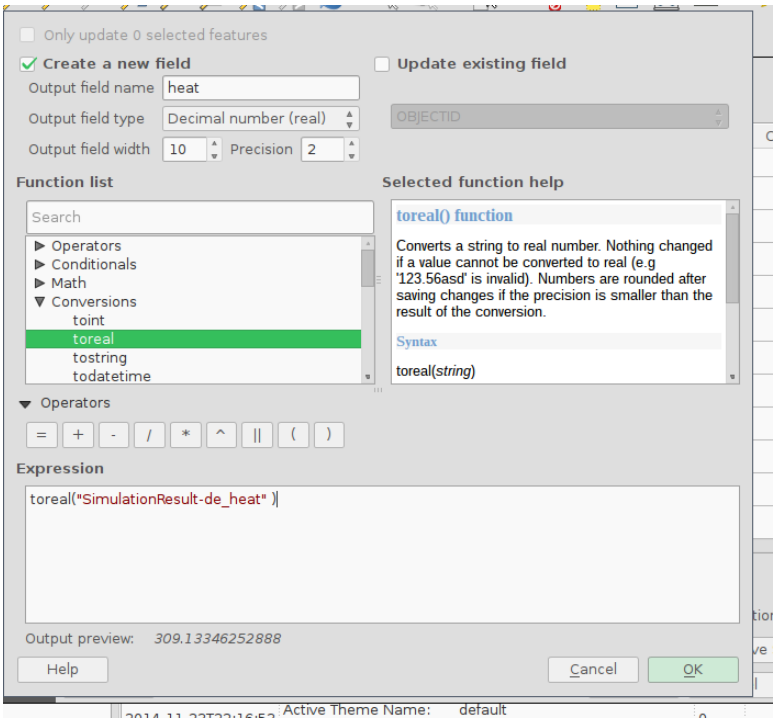

Figure : *Screenshot*

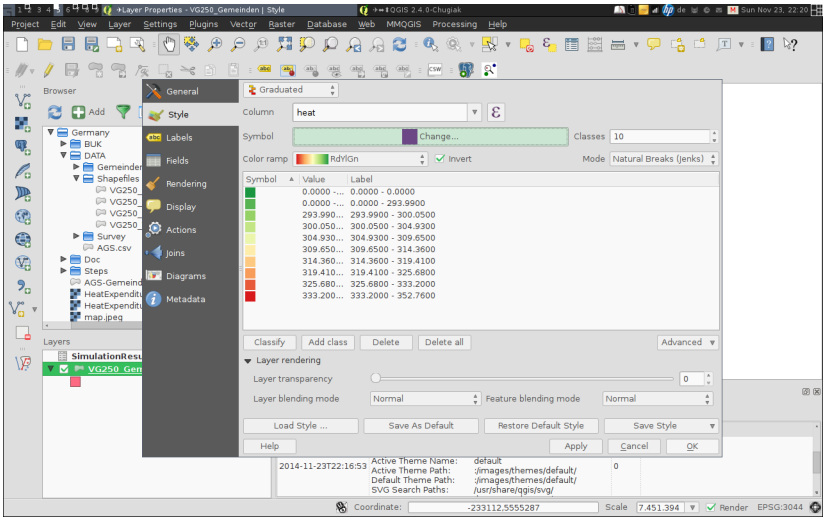

Figure : *Screenshot*

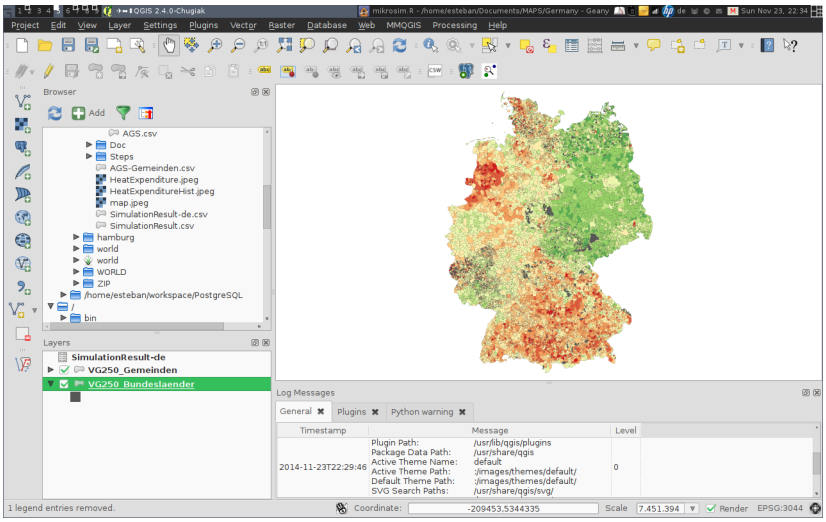

Figure : *Screenshot*

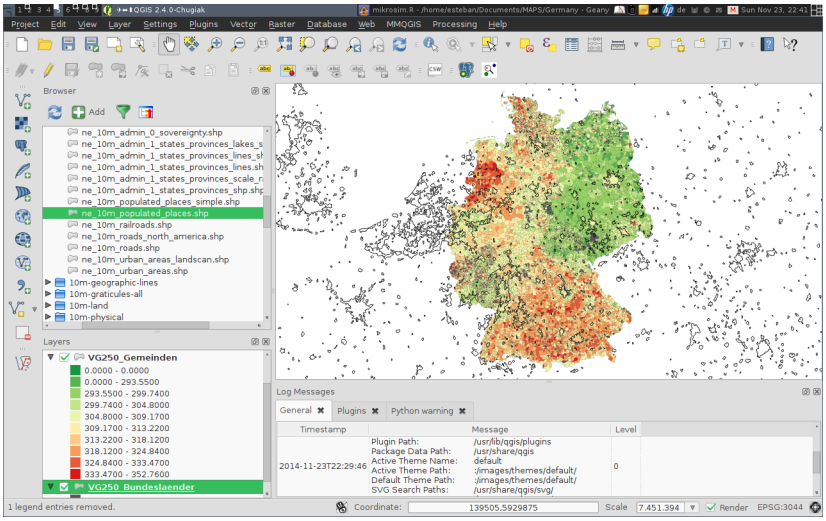

Figure : *Screenshot*

Simulation Results

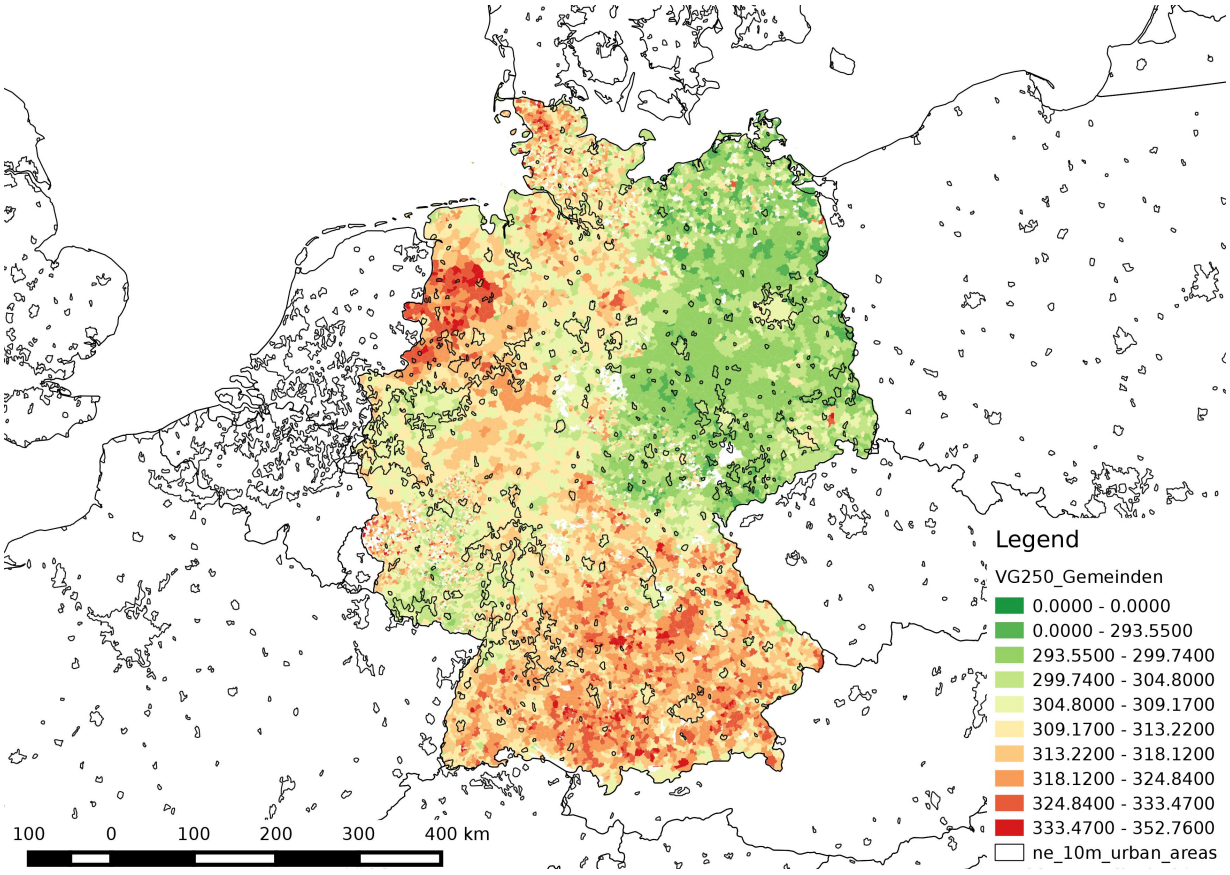

Figure 1: *Map showing the simulation result for all German states.*

Heat expediture for the "North Rhine-Westphalia" federal state

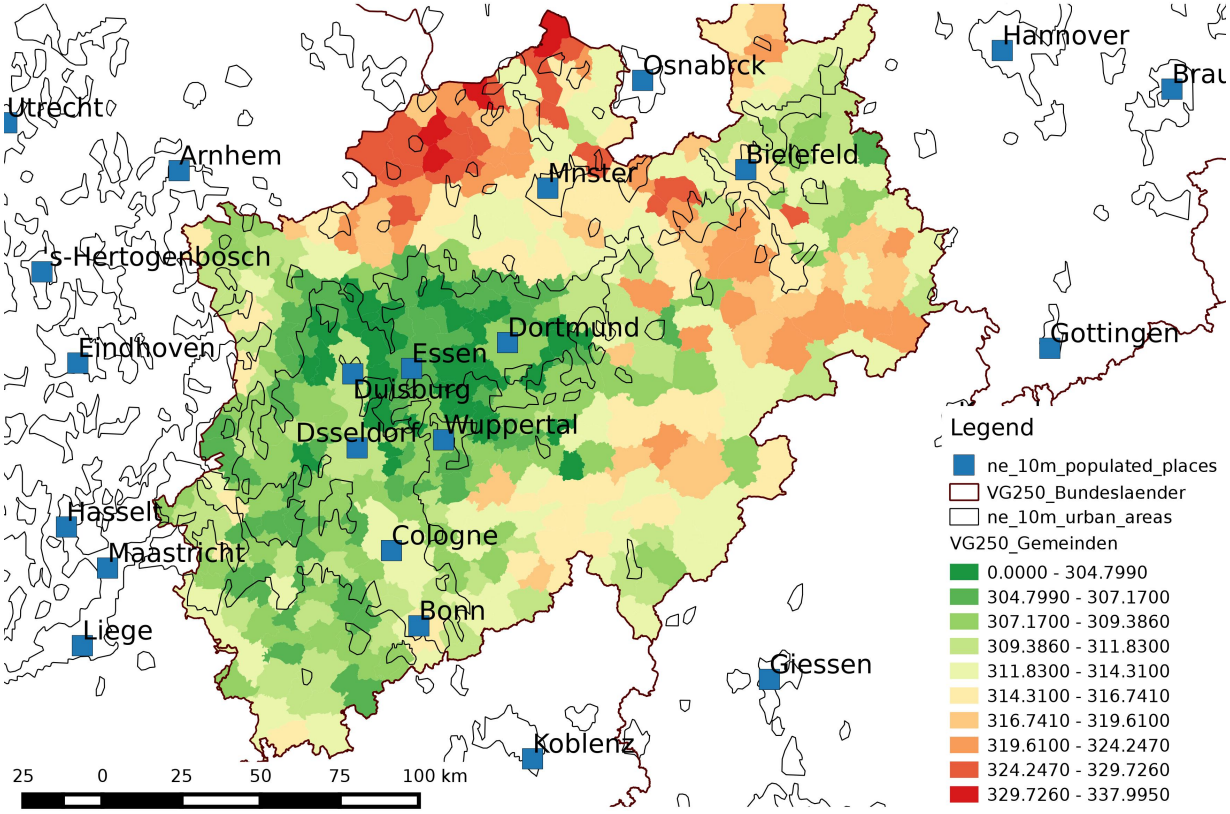

Figure 1: *Map showing the result from our simulation for the state of North Rhine-Westphalia.*

Annex: Scripts to fetch the required data from the internet

For the purpose of this workshop I prepare a zip file containing all the required data to run the simulation. If you decompress the folder you will automatically get the folder structure describe in section Getting the data and scripts to run the simulation

You can download the file under the following link.

http://asampleurl.com

If you want to download the raw data used in the simulation just go to the links described below. You can also find complementary information regarding this data in these web pages.

- **The data from the census can be found under the following link**

  https://ergebnisse.zensus2011.de

- **The micro census as csv file**

  http://www.forschungsdatenzentrum.de/bestand/mikrozensus/cf/2002/fdz_mikrozensus_cf_2002_ascii-csv.zip

- **The shapefiles for visualization**

  https://www.zensus2011.de/SharedDocs/Downloads/DE/Shapefile/VG250_1Jan2011_UTM32.zip?__blob=publ

Some important pieces from the file 'cleanAGS.py' are listed below.

In order to download the data from the web page I generate a url with the area codes and the attribute I want to get. The web page has a restriction on queering a maximum of 100 areas at the same time and therefor I have to pass 100 area codes at a time. I will download each csv file and latter on combine them on a single file.

```
# divide the area codes in chunks of 100 items
dat_AGS = chunks(AGS, 100)
# iterate through all chunks
for num, ags_c in enumerate(dat_AGS):
    # format the download link
    to_download = DOWNLOAD_LINK.format(ags_id=ags_c, constrain=constrain)
    # remove blank spaces from the url
    to_download = to_download.replace(" ", "")
    # create a name for the csv file
    download_name = "./DATA/Gemeinden/{}-{}.csv".format(constrain, num)
    # fetch the csv file
    url.urlretrieve(to_download, filename=download_name)
    # wait 1 second to get the next file
    sleep(1)
return(num)
```

This piece of code reads the area codes and selects only the desire codes (Gemeinden). The first line of code defines which file to open, the separator character, the boolean value *None* to tell python that the file does not have any header and finally define the name of the imported columns so that we can access the data using this names. The second line select only areas corresponding to the codes of the desire level (Gemeinden). Code lines 3 to 6 modify the code for some areas, these areas are both "Gemeinde" and "State" and therefor have many codes representing the same area. This is important for the latter visualization because the QGIS will not be able to identify this area codes.

```
_ags = pd.read_csv("./AGS.csv", sep="\t", header=None, names=['ags', 'name'])
_clean_ags = _ags[_ags['ags'] >= 10000000000]
```

```
3 _clean_ags[_clean_ags['ags'] == 20000000000] = 2     # Hamburg
4 _clean_ags[_clean_ags['ags'] == 40110000000] = 4011  # Bremen
5 _clean_ags[_clean_ags['ags'] == 40120000000] = 4012  # Bremerhaven
6 _clean_ags[_clean_ags['ags'] == 110000000000] = 11   # Berlin
7 _clean_ags.to_csv("./AGS-Gemeinden.csv")
8 AGS = _clean_ags['ags'].tolist()
```

This piece of code reads the downloaded data from the census 2011. The first command describe how to read the csv file. It describes:

1. The file name
2. The value separator **(;)**
3. Number of header lines
4. Character defining **NA** values
5. Number of lines at the end of the file
6. Engine to be used to read the file. Normally we will use a C engine, as this is faster, unfortunately the implementation of this engine doesn't have a notion of footer lines
7. Which column should we use as index
8. Encoding of the file, important if we have a file with non standard characters.

The rest of this lines remove unwanted characters from the records. Some record ere within brackets, indicating a manipulation in the value to avoid the identification of individuals. The area codes in this data set do not only contain the numerical code but also the name of the area, I removed all non numerical characters from the area code in order to work with them.

```
 1 # read the csv file
 2 data = pd.read_csv(
 3     file_name, sep=";", header=5, na_values="-",
 4     skip_footer=7, engine='python', index_col=index,
 5     encoding="latin-1")
 6
 7 # transpose the data
 8 data = data.transpose()
 9
10 # reformat the data index
11 new_index = data.index
12 # delete all non numeric characters
13 new_index = new_index.map(lambda x: re.sub('[^0-9]', '', x))
14 # update the index
15 data.set_index(new_index, inplace=True)
16
17 # some records are within brackets, remove them
18 for col in data.columns:
19     data[col] = data[col].map(lambda x: str(x).lstrip('(').rstrip(')'))
20
21 return(data)
```

1 http://cran.r-project.org/web/packages/MASS/
2 https://www.zensus2011.de
3 http://www.forschungsdatenzentrum.de
4 http://qgis.org