R as a tool for visualization of large demographic datasets

Anna Dmowska

Institute of Geoecology and Geoinformation,
Adam Mickiewicz University, Poznan
dmowska@amu.edu.pl, http://dmowska.home.amu.edu.pl

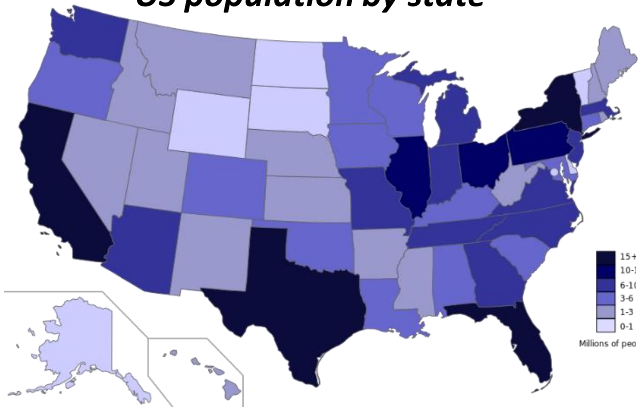
Why R 2018, 2.07-5.07.2018, Wrocław

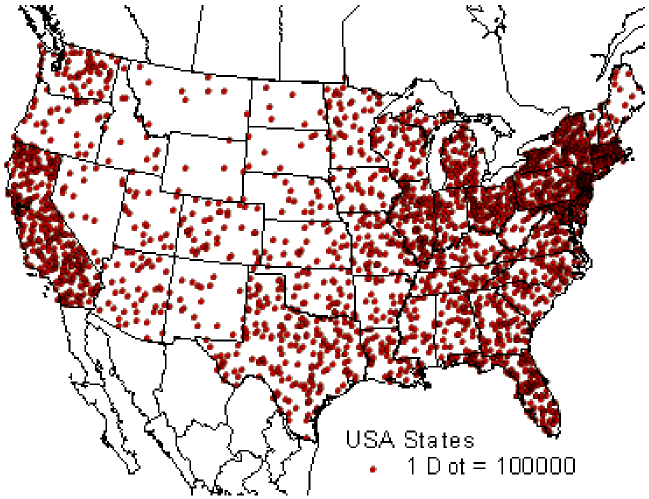
Demographic data

- aggregated over previously defined regions (e.g., counties, tract, statistical units)
- stored as tabular data
- visualized by assigning one color to whole aggregated units (choropleth map) or by using dot density maps.

State	Population
California	37 253 956
Texas	25 145 561
New York	19 378 102
Florida	18 801 310
Illinois	12 830 632
Pensylwania	12 702 379
Ohio	11 536 504
Michigan	9 883 640
Georgia	9 687 653
North Carolina	9 535 483

US population by state

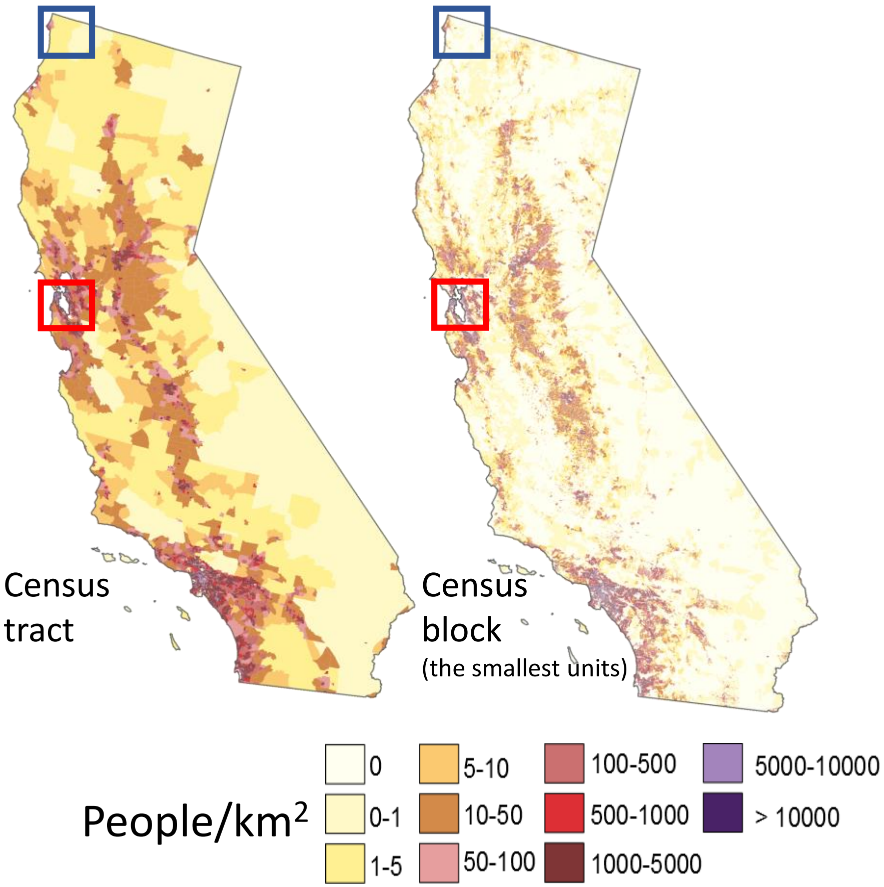




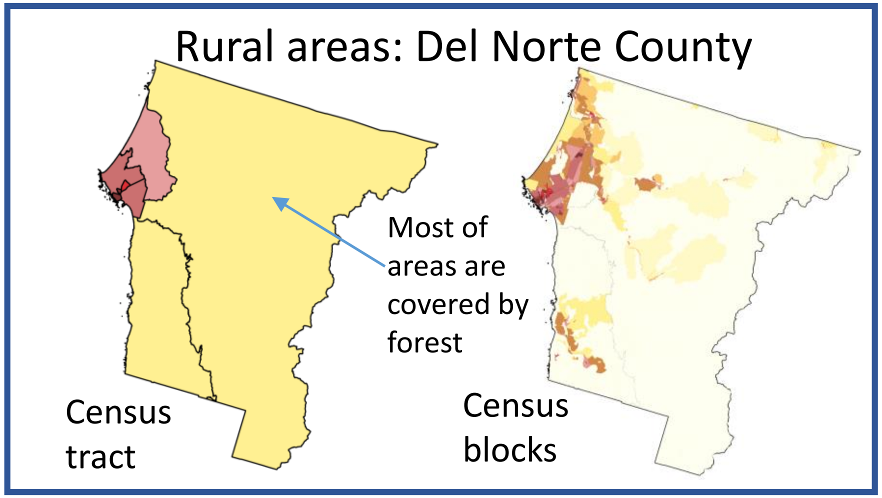
http://maegansmaps.blogspot.com

Demographic data

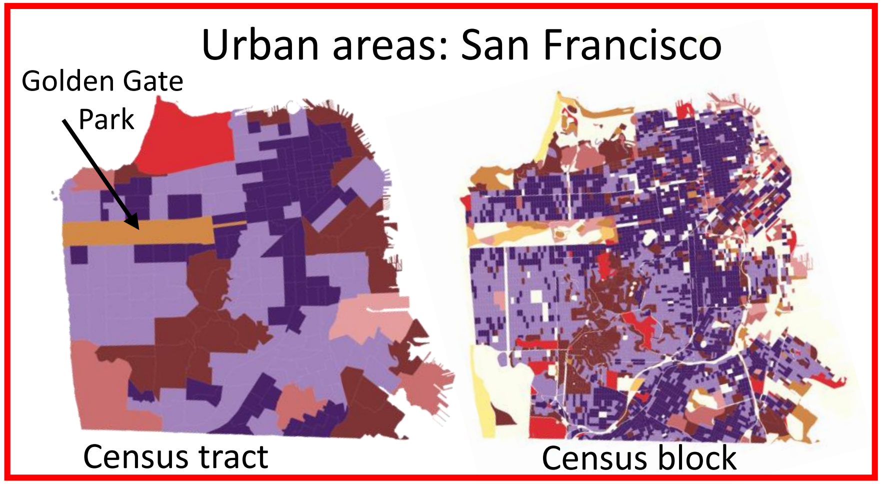
State of California



Rural areas: Del Norte County



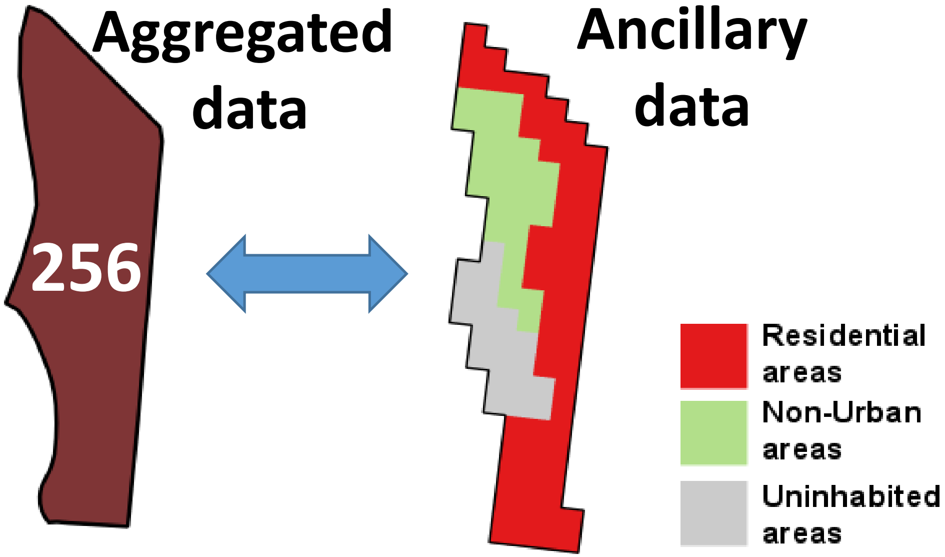
Urban areas: San Francisco



DATA AGGREGATED TO UNITS
Spatial resolution dependent on the choice of Census units and spatially varying; lower in rural areas, higher in urban areas
Mapped population is **distributed uniformly** within each Census unit
The extents of **Census units change with time**, which makes difficult year-to-year comparison

From aggregated data into hi-res grid

Dasymetric modeling refers to a process of **disaggregating spatial data to a finer unit** of analysis, **using additional** (or **ancillary**) **data** to help refine locations of population or other phenomena (Mennis 2003).



Aggregated data

256

Ancillary data

Residential areas

Non-Urban areas

Uninhabited areas

A single aggregaition spatial units

Race	#people
White	153
Black	18
Asian	78
Other	4
Hispanic	3

People/km^2

0

0-1

1-5

5-10

10-50

50-100

100-500

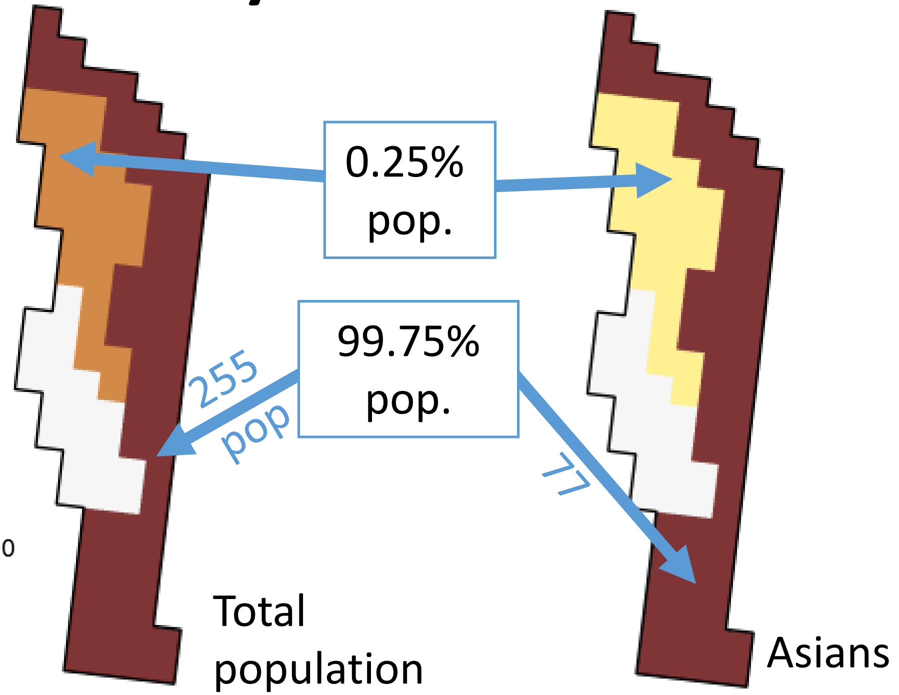
500-1000

1000-5000

5000-10000

> 10000

Dasymetric model



0.25% pop.

99.75% pop.

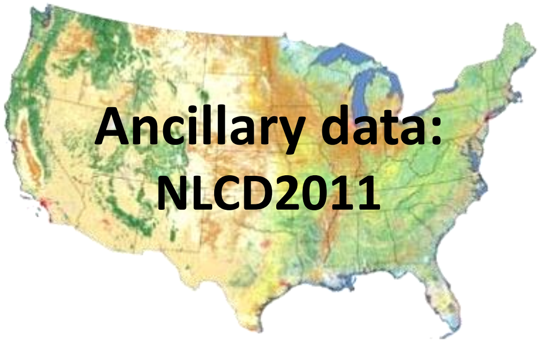
255 pop

77

Total population

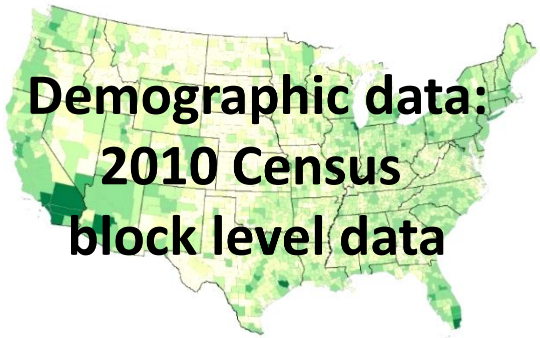
Asians

Dasymetric modeling for large areas



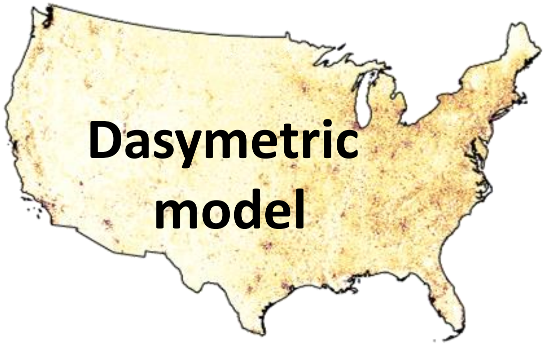
Ancillary data:
NLCD2011

8 651 173 750 cells



Demographic data:
2010 Census
block level data

#blocks: 11,15 milions



Dasymetric
model

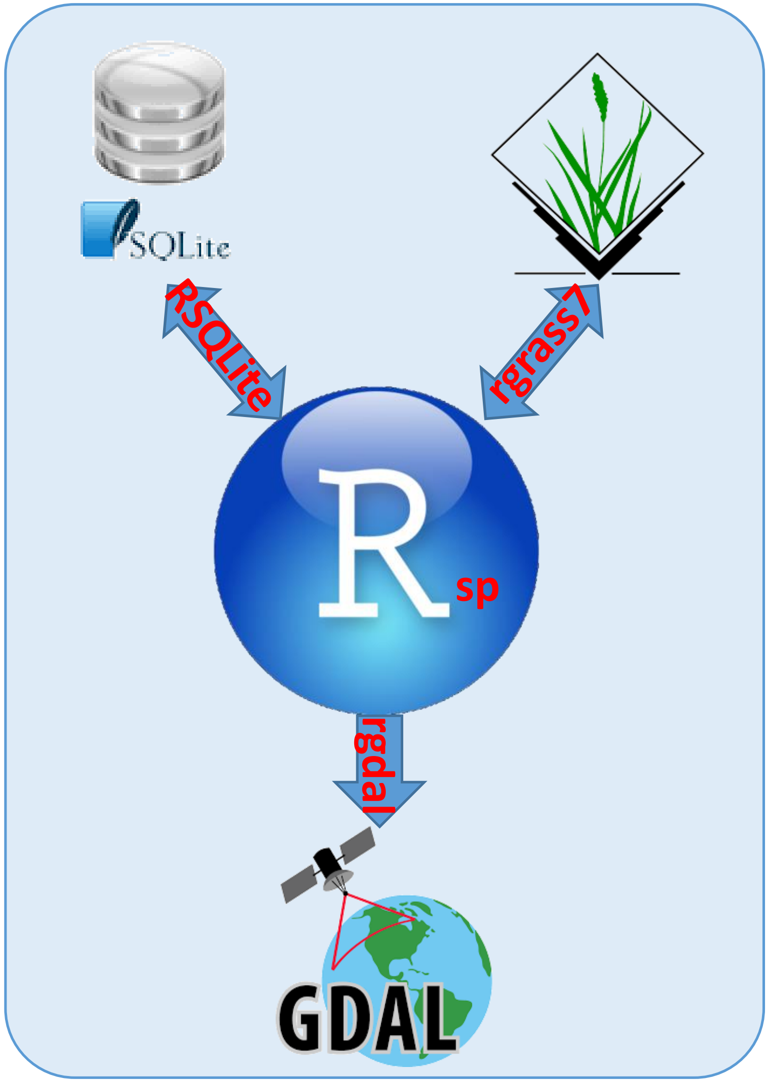
8 651 173 750 cells

Calculation time (for one map)

55 hours

The need to develop an **efficient, fully automated algorithm** to work with large datasets, which will allow to perform calculations within a reasonable time

Dasymetric modeling for large areas



Allow to build **efficient**, **flexible** and **fully automated** computational environment to work with large dataset **without advanced programming** skills.

R is a comprehensive computational environment that includes **libraries to work with different types of data**: ***geospatial data*** (sp, rgrass7, raster, rgdal), ***standard relational databases*** (DBI, RSQLite).

Main advantages of using R over GIS software are: less processing steps are required, no intermediate layers, increased flexibility and automation

scripts and sample data are available at
http://sil.uc.edu

Handling large dataset in R

1 To manage data storage requirements and to better control the time of computation we **divide U.S. into separate counties**.

- We used region concept in GRASS GIS for computationally efficient division of U.S. into separate counties.

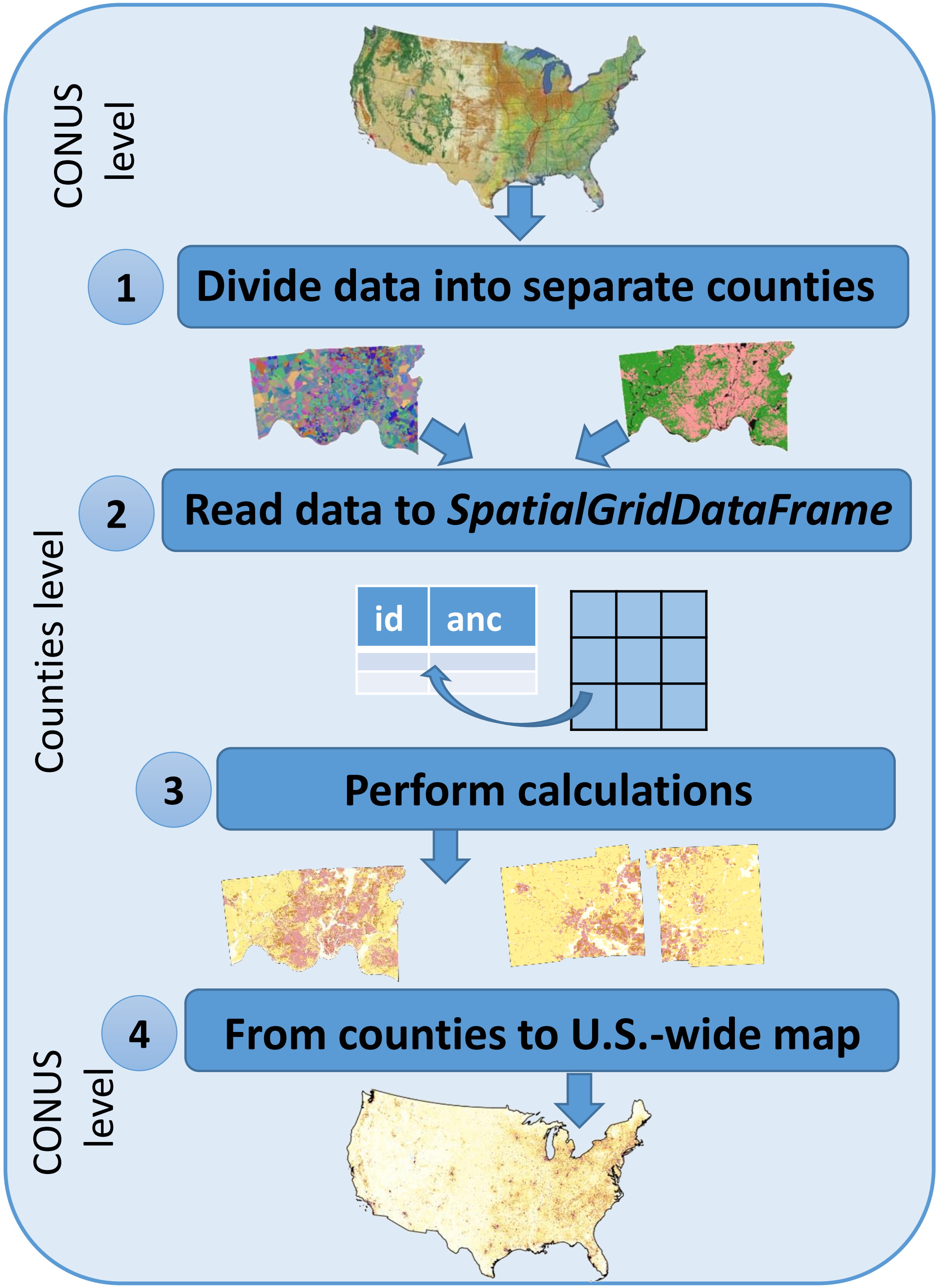
2 Raster **data for each county is read into *SpatialGridDataFrame* object in R**

- This structure allow to integrate information about its spatial content with Census data into a single relational model.

3 We **process each county separately**.

4 In the last step **maps for individual counties are joined into U.S.-wide map**

CONUS=conterminous U.S.



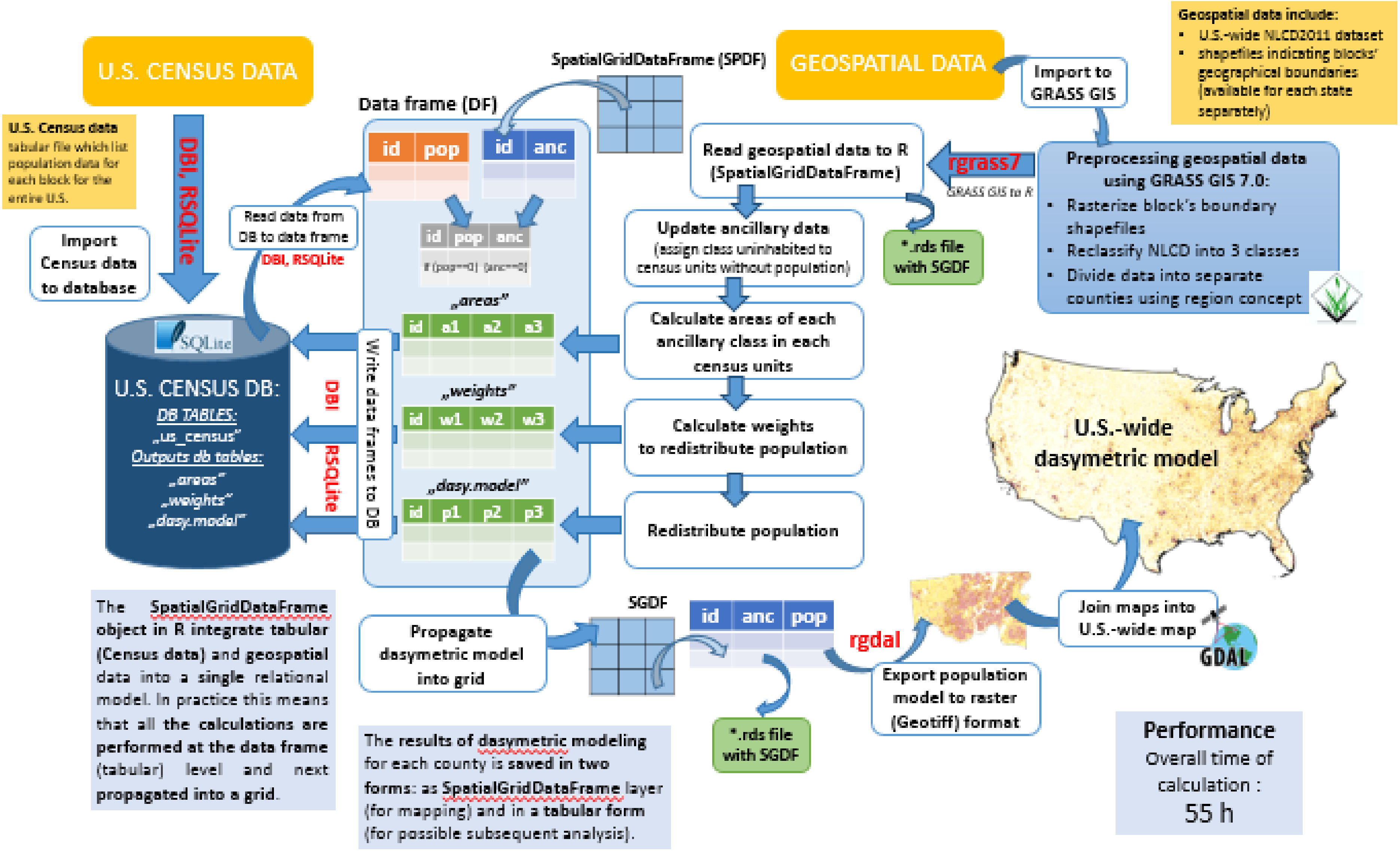
1 Divide data into separate counties

2 Read data to *SpatialGridDataFrame*

3 Perform calculations

4 From counties to U.S.-wide map

How our algorithm works?



U.S. Census data tabular file which list population data for each block for the entire U.S.

Geospatial data include:

- U.S.-wide NLCD2011 dataset
- shapefiles indicating blocks' geographical boundaries (available for each state separately)

Preprocessing geospatial data using GRASS GIS 7.0:

- Rasterize block's boundary shapefiles
- Reclassify NLCD into 3 classes
- Divide data into separate counties using region concept

Read geospatial data to R (SpatialGridDataFrame)

Update ancillary data (assign class uninhabited to census units without population)

Calculate areas of each ancillary class in each census units

Calculate weights to redistribute population

Redistribute population

Propagate dasymetric model into grid

Export population model to raster (Geotiff) format

Join maps into U.S.-wide map

The SpatialGridDataFrame object in R integrate tabular (Census data) and geospatial data into a single relational model. In practice this means that all the calculations are performed at the data frame (tabular) level and next propagated into a grid.

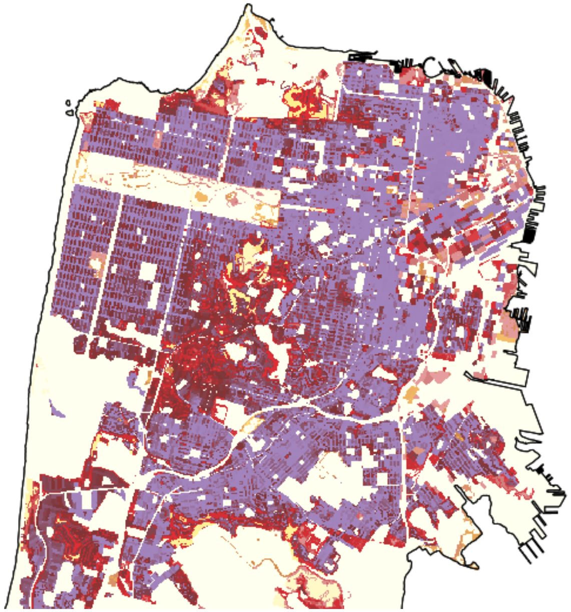
The results of dasymetric modeling for each county is saved in two forms: as SpatialGridDataFrame layer (for mapping) and in a tabular form (for possible subsequent analysis).

Performance

Overall time of calculation :

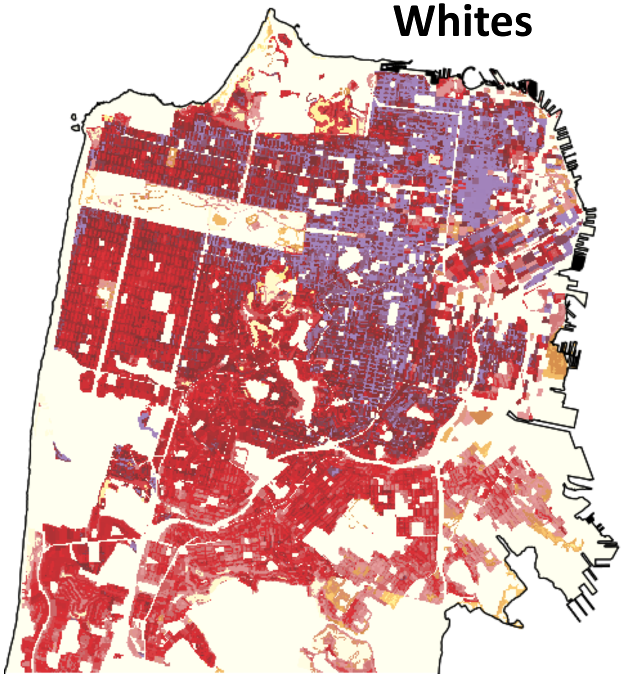
55 h

Distribution of racial/ethnicity

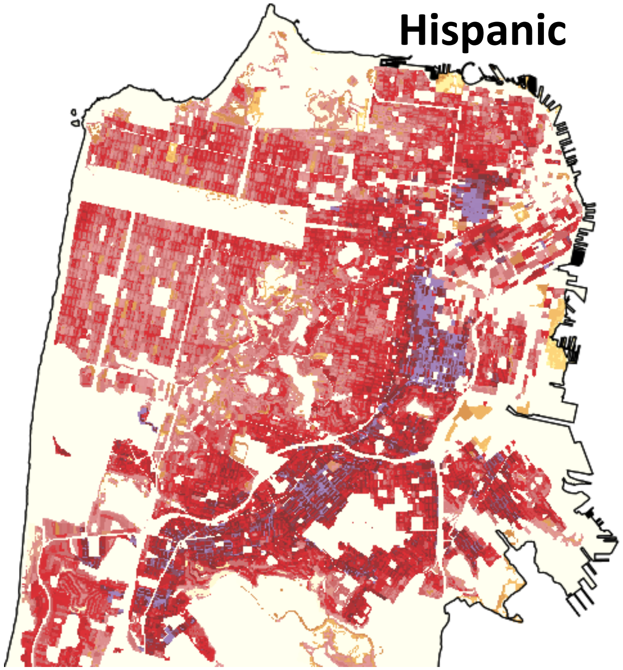


Total population

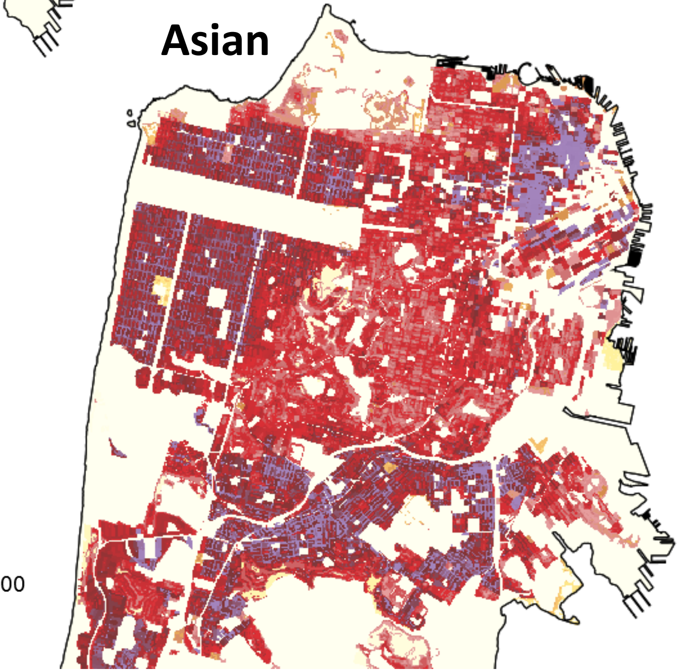
San Francisco, CA



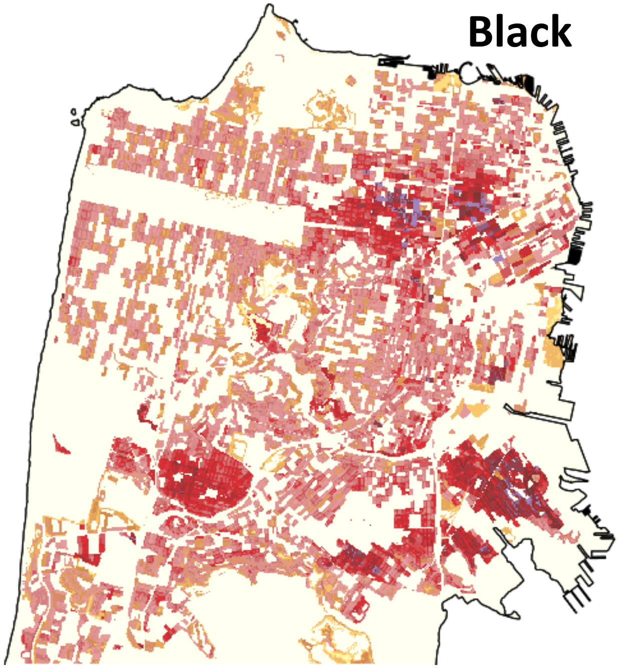
Whites



Hispanic



Asian

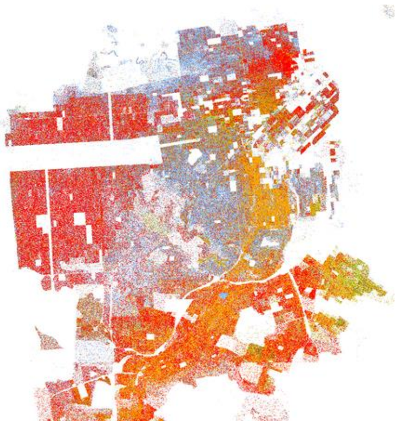


Black

People/km^2

□ 0	■ 5-10	■ 100-500	■ 5000-10000
■ 0-1	■ 10-50	■ 500-1000	■ > 10000
■ 1-5	■ 50-100	■ 1000-5000	

Dot density maps



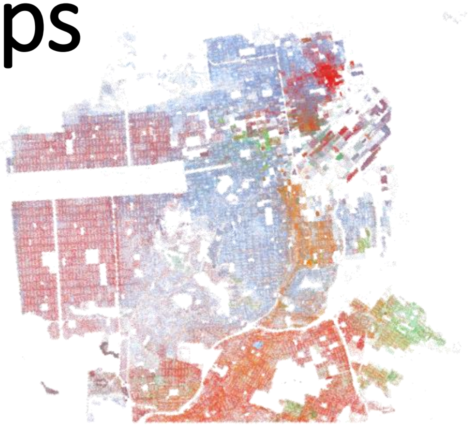
🔵 White

🟢 Black

🔴 Asian

🟠 Hispanic

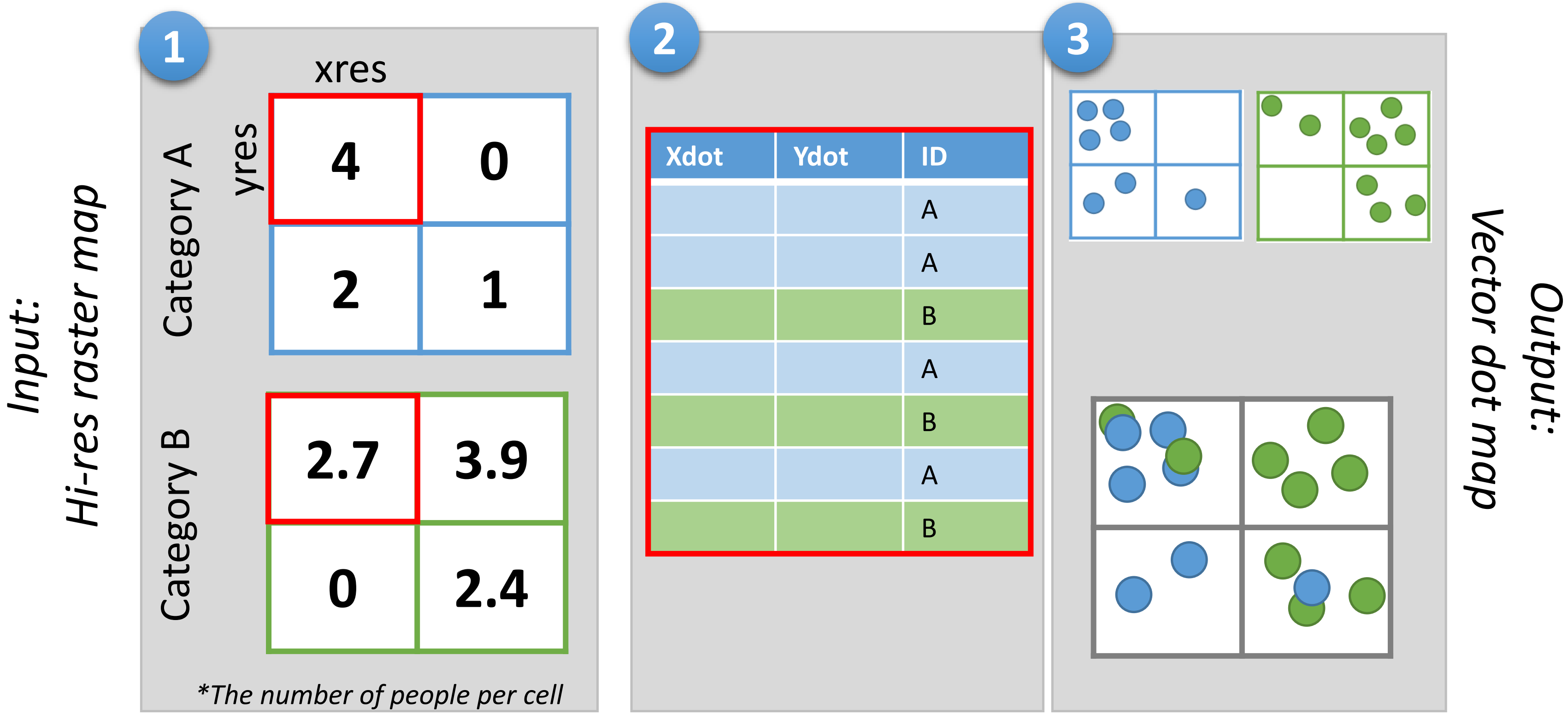
🟤 Other race/
Native American



http://demographics.coopercenter.org/DotMap/

	Standard algorithm	Proposed algorithm
Input data	created based on **data aggregated** to predefined regions.	created based on **high resolution raster** data (i.e the results of dasymetric modeling)
	dots are randonly distributed in each region	dots are randomly distributed in each cell.
R tools	maptools::dotsInPolys	so far **no algorithms** to produce dot density maps using raster data.
	works only for large polygons and it is inefficient	works with high resolution maps (even for cells 30x30m)
Visualization	a predetermined order of displaying racial groups • i.e white at the bottom, then black, Asian, Hispanic (visual effect: more Asians than white).	it builds random stack where probability of displaying a point at the top depends on the percentage of the race in a cell.

How it works?



Input:
Hi-res raster map

1

xres

yres

Category A

4	0
2	1

Category B

2.7	3.9
0	2.4

*The number of people per cell

2

Xdot	Ydot	ID
		A
		A
		B
		A
		B
		A
		B

3

Output:
Vector dot map

1 Algorithm uses raster data as an input and dots are randomly scattered in each cell. In situation when approximate number of people for one cell is below 1 algorithm uses probabilistic approach to decide whether to place a point in a given cell or not.

2 If visualization cover more than one race it also build random stack where probability of displaying a point at the top depends on the percentage of the race in a cell.

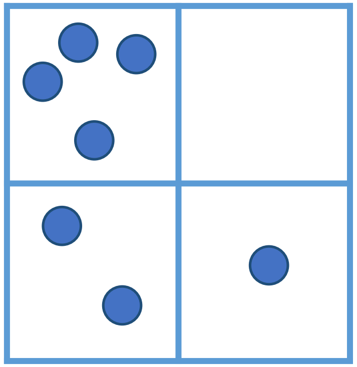
3 The results is a vector dot map.

How it works?

Aggregated population data

4	0
2	1

Hi-res raster population data

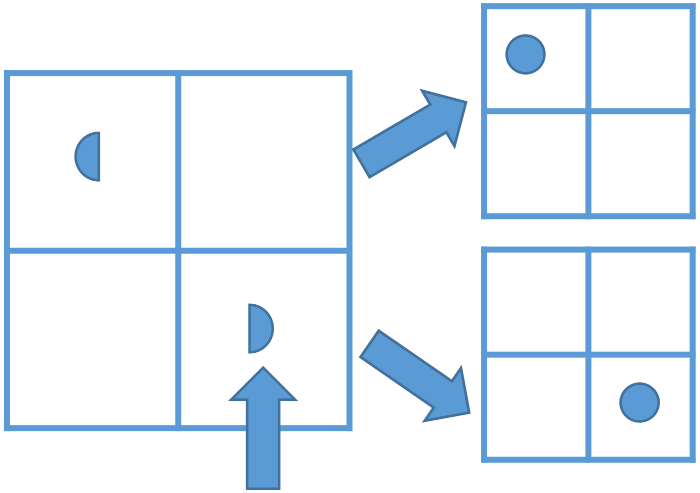


Dot map

1 dot = 1 person

0.5	0
0	0.5

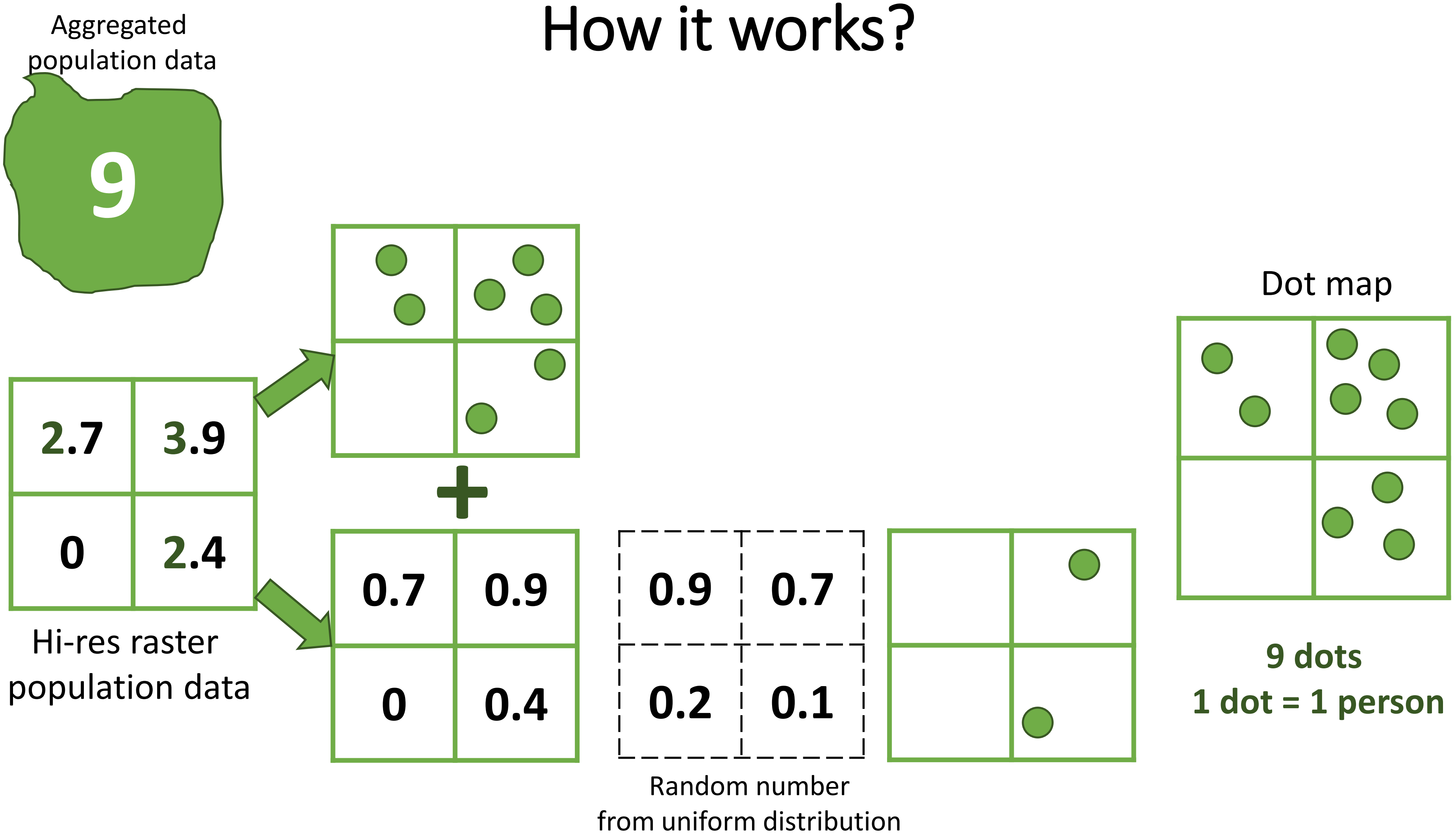
Using high resolution raster map approximate number of people for one cell can be a fraction.



We cannot place half a dot.

Where live this person?

How it works?



Aggregated population data

9

2.7	3.9
0	2.4

Hi-res raster population data

+

0.7	0.9
0	0.4

0.9	0.7
0.2	0.1

Random number from uniform distribution

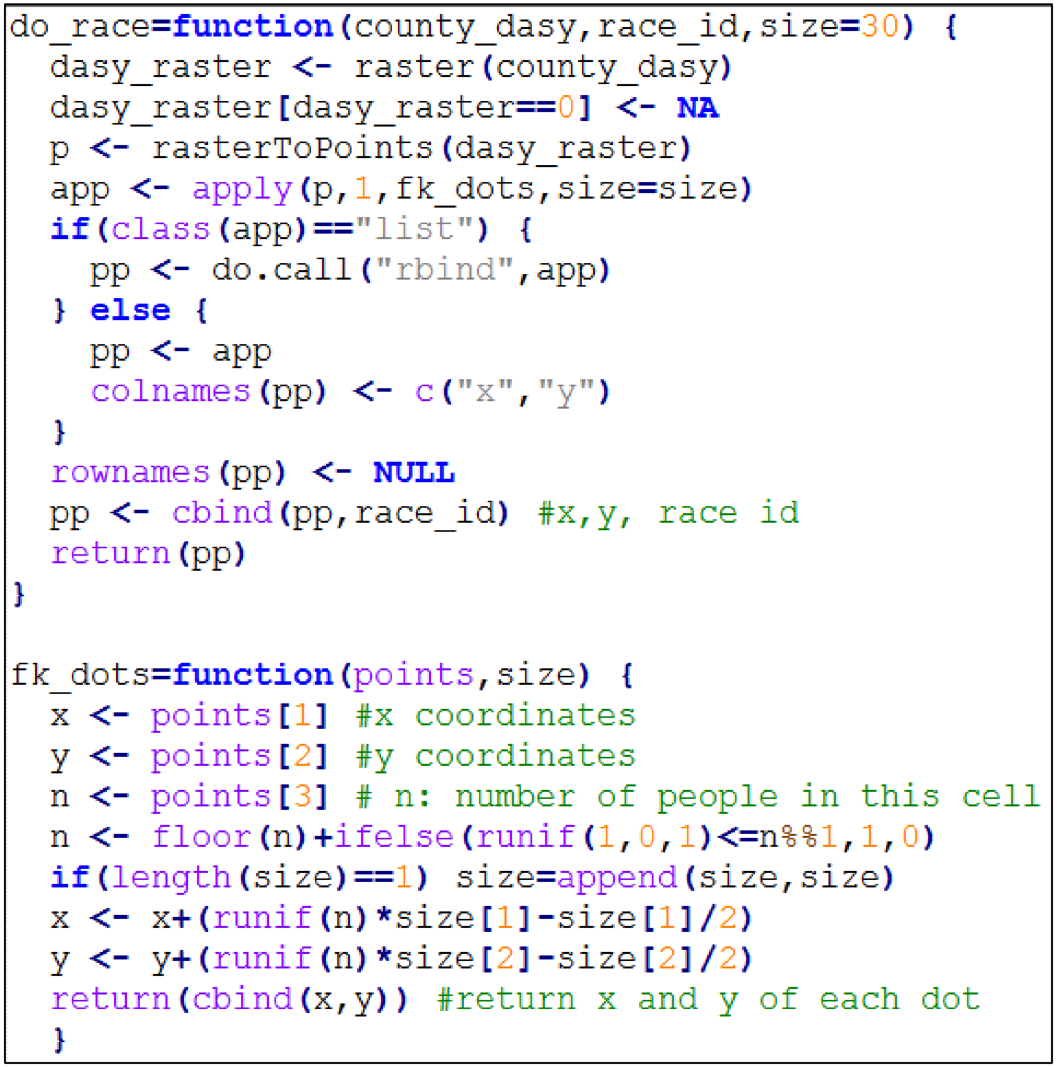
Dot map

9 dots

1 dot = 1 person

- If the number of people per cell is below 1 algorithm uses probabilistic approach to decide whether to place a point in a given cell or not.
- For each cell is drawn the number from 0 to 1 using uniform distribution.
  - Number in cell > drawn number – dot is placed in this cell

R implementation



```
do_race=function(county_dasy,race_id,size=30) {
  dasy_raster <- raster(county_dasy)
  dasy_raster[dasy_raster==0] <- NA
  p <- rasterToPoints(dasy_raster)
  app <- apply(p,1,fk_dots,size=size)
  if(class(app)=="list") {
    pp <- do.call("rbind",app)
  } else {
    pp <- app
    colnames(pp) <- c("x","y")
  }
  rownames(pp) <- NULL
  pp <- cbind(pp,race_id) #x,y, race id
  return(pp)
}

fk_dots=function(points,size) {
  x <- points[1] #x coordinates
  y <- points[2] #y coordinates
  n <- points[3] # n: number of people in this cell
  n <- floor(n)+ifelse(runif(1,0,1)<=n%%1,1,0)
  if(length(size)==1) size=append(size,size)
  x <- x+(runif(n)*size[1]-size[1]/2)
  y <- y+(runif(n)*size[2]-size[2]/2)
  return(cbind(x,y)) #return x and y of each dot
  }
```

function ***do_race*** take 3 arguments:

- county_dasy – the result of dasymetric modeling stored as SpatialGridDataFrame
- race_id – race/ethnicity group id
- size – cell resolution

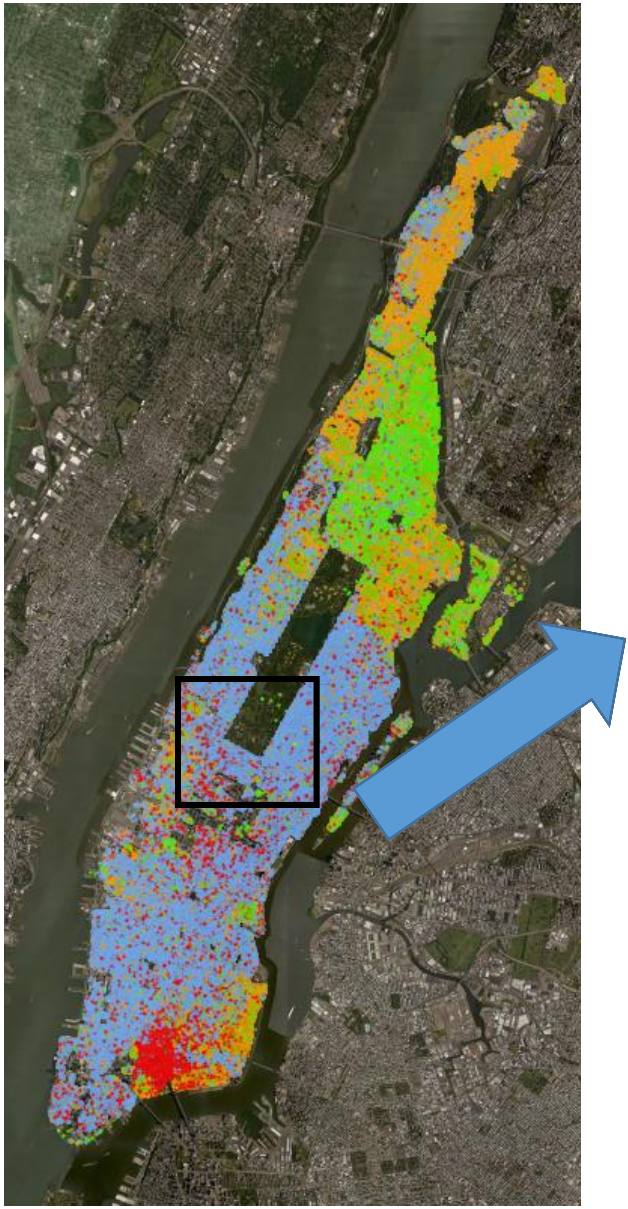
This function return a matrix with coordinates of each dot (x, y) and race id.

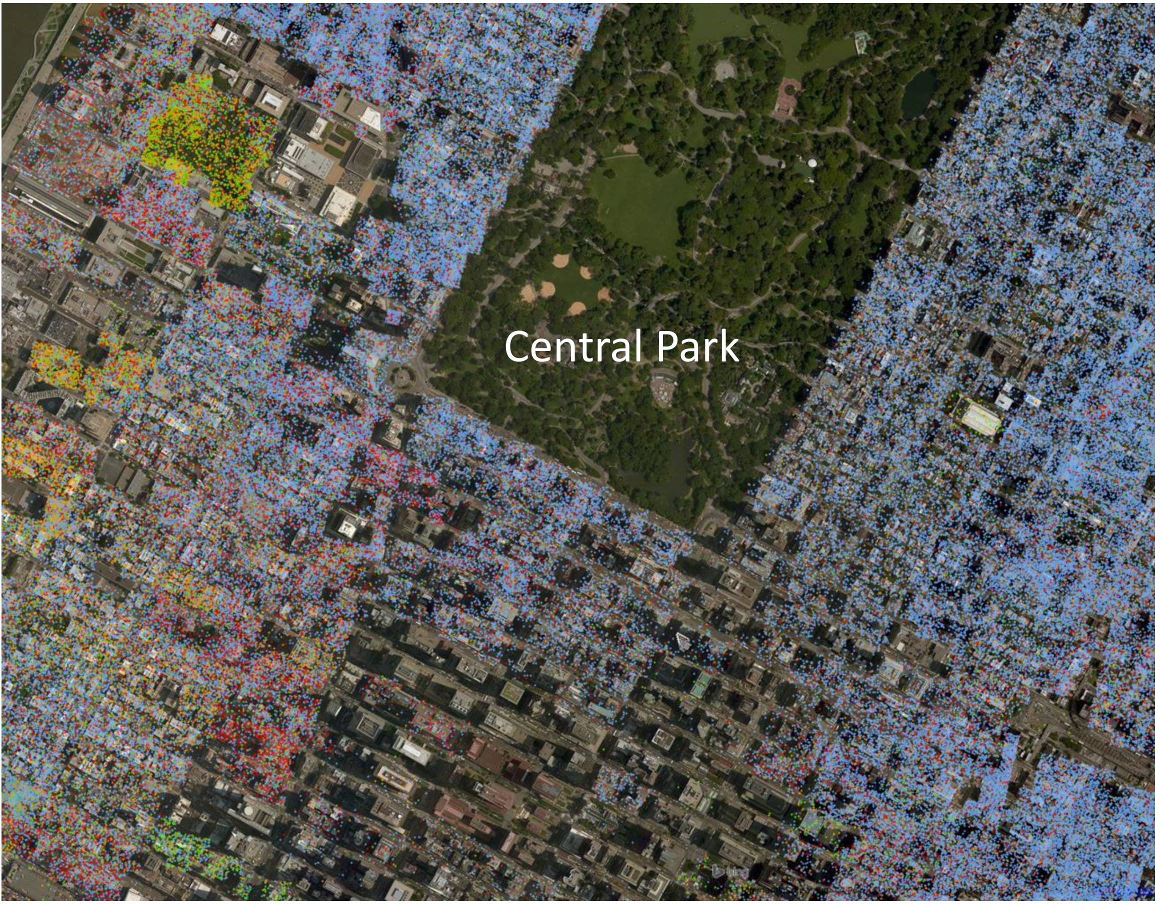
The main part of this funtion is:

apply(p,1,fk_dots,size=size)

- **p** is an output from rasterToPoints() conversion, there is a matrix with x,y, nb_of_pop, each row contain the data for one cell.
- **fk_dots** is a function which generate number of dots for each cell and return a matrix with x,y coordinates of each dot. size is a size of raster cell; this function will be apply for each row in object p
- **size** is an argument given to function fk_dots; there is a cell resolution.

Examples: Manhatan, New York





1 dot = 1 person

Population density: 27799/km^2

● White

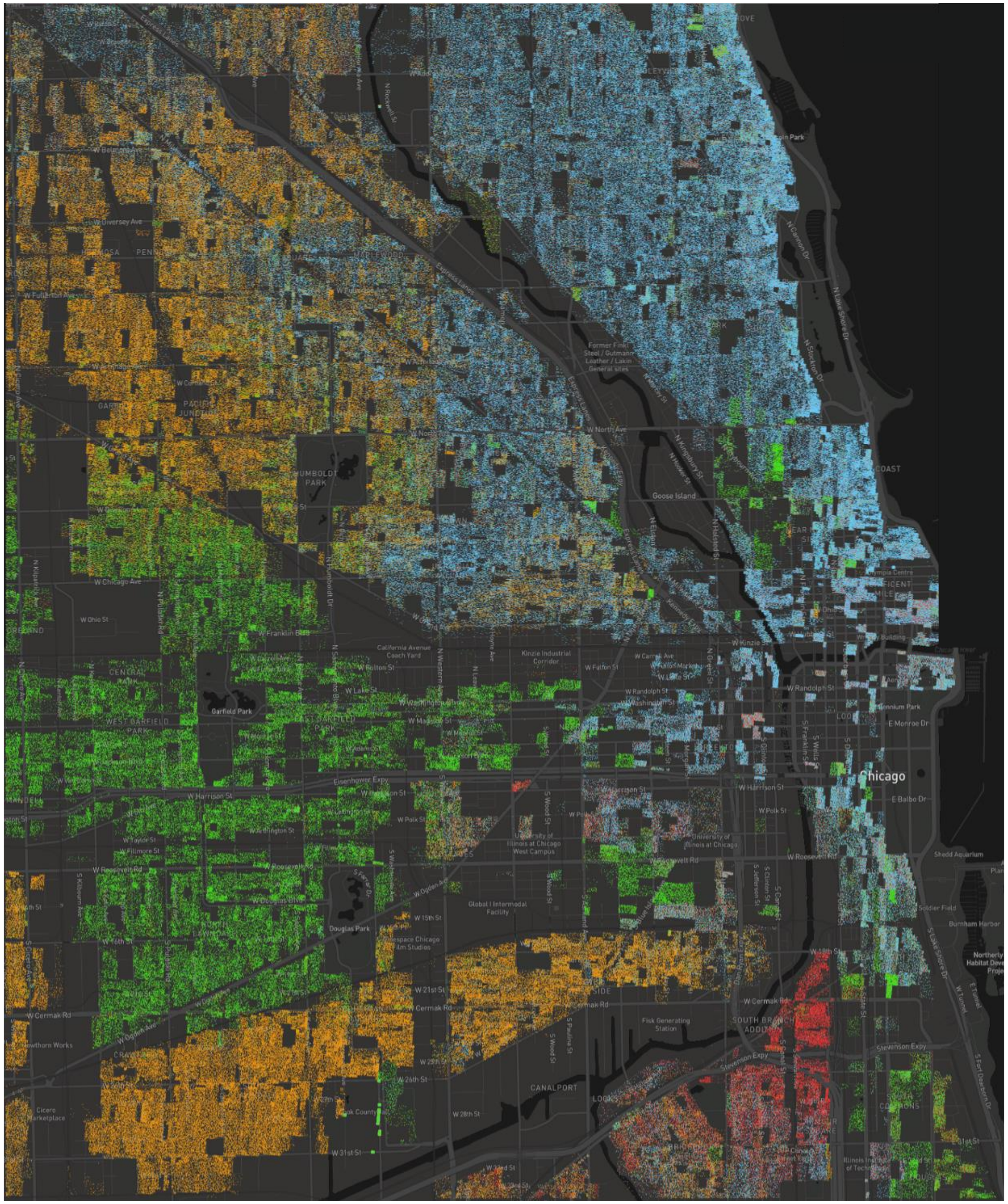
● Black

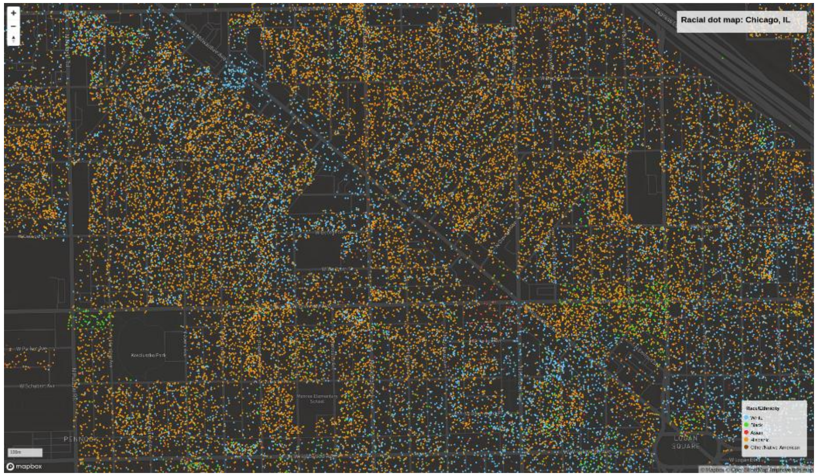
● Asian

● Hispanic

● Other Race / Native American / Multi-racial

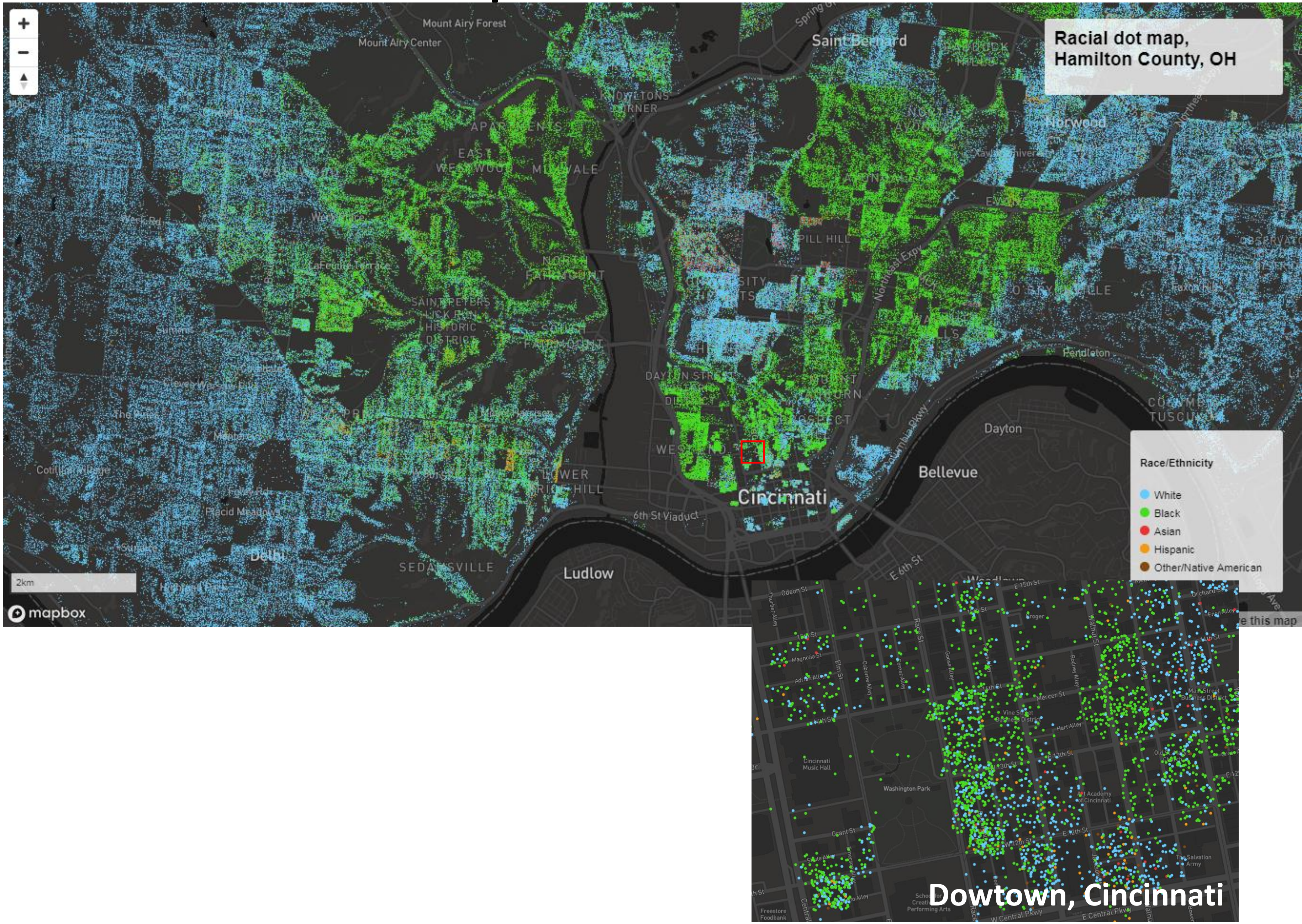
Examples: Chicago, IL





„Jackowo”

Examples: Cincinnati



Racial dot map,
Hamilton County, OH

Dowtown, Cincinnati

Thank you

Hi-res maps are available here

http://sil.uc.edu/webapps/socscape_usa/