

2IV35 Visualization

Assignment 1

Information Visualization

Yiting Xu (0925537)
Shah Nawaz (0927948)
{y.xu, s.nawaz.1}@student.tue.nl

Dec. 7, 2014

TABLE OF CONTENTS

| | |
|--|-----------|
| ABSTRACT..... | 3 |
| 1. INTRODUCTION | 4 |
| 2. TOOL: D3..... | 5 |
| 3. DATA SET | 6 |
| 4. TASK 1: FOREIGNERS PROPORTION OF CITIES IN THE NETHERLANDS..... | 7 |
| 4.1. TASK..... | 7 |
| 4.2. MOTIVATION..... | 7 |
| 4.3. TECHNICAL DESIGN..... | 7 |
| 4.4. IMPLEMENTATION..... | 8 |
| 5. TASK 2: GENERAL MARITAL STATUS IN THE NETHERLANDS..... | 13 |
| 5.1. TASK..... | 13 |
| 5.2. MOTIVATION..... | 13 |
| 5.3. TECHNICAL DESIGN..... | 13 |
| 5.4. IMPLEMENTATION..... | 14 |
| 6. TASK 3: AGE & GENDER DISTRIBUTION OF CITIES IN THE NETHERLANDS | 16 |
| 6.1. TASK..... | 16 |
| 6.2. MOTIVATION..... | 16 |
| 6.3. TECHNICAL DESIGN..... | 16 |
| 6.4. IMPLEMENTATION..... | 17 |
| 7. TESTS | 21 |
| 7.1. BROWSERS AND ENVIRONMENT | 21 |
| 7.2. TASK 1 | 21 |
| 7.3. TASK 2 | 21 |
| 7.4. TASK 3 | 22 |
| 8. CONCLUSIONS..... | 23 |
| REFERENCE..... | 24 |
| APPENDIX | 25 |

Abstract

Given by the dataset of shape information of Dutch cities, we solved three tasks by using three different visualization techniques in a visualized and interactive way. Because of the large-scale data, we choose to use Data-Driven Documents (D3), which is a powerful JavaScript library for manipulating documents based on data. For each task, we find a correspondingly suitable technique according to its task requirements, and add some interesting interactions. In this report, we will not only describe the design and implementation process, but also the tests and results. We find that the information visualization really helps to understand the data easier.

Keywords

Visualization, Interaction, D3, JavaScript, Dutch cities

1. Introduction

Data analysis is an indispensable part of all applied research and problem solving in industry. The most fundamental data analysis approaches are visualization, including techniques like histograms, scatter plots, surface plots, tree maps, parallel coordinate plots, etc. [1]. With the development of visualization in computer science, information visualization helps people view the data in a straightforward and even interactive way. In the era of large-scale data, visualization really improves people to understand the data easily, fast and directly.

Three tasks are formulated by the given data set of shape information of Dutch cities, and we choose to use D3 [2] to solve them in a visualized and interactive way. The decision to use D3 is based on the fact that user would be able to visualize the dataset easily using their own web browser without needing to install additional software. Detailed information of this library can be found in section 2. We chose three different techniques for each of the tasks, which are basically point-based, line-based and region-based techniques. The choice of techniques corresponds with the requirements of tasks and the provided dataset.

The compressed code attached to this report contains 4 html files and 3 other data files. The 4 html files are indexed by the index.html (see figure 1), in which user can click the 'CLICK' and see the 3 tasks and their final visualized graphs. In each task html, there is one 'BACK' link on the top-left corner to return to the index page. The data files contains one GEOJson file, one .txt file and one .tsv file, more information about the data files and dataset can be found in section 3 and Appendix.

| 2IV35 Visualization Assignment 1 | Table of Contents |
|--|--|
| Yiting Xu (0925537) y.xu@student.tue.nl | Task 1 Foreigners Proportion of Cities in Netherlands CLICK Interactive Scatterplot (Point-based Technique) |
| Shah Nawaz (0927948) s.nawaz.1@student.tue.nl | Task 2 General Marital Status in Netherlands CLICK Parallel Coordinates (Line-based Technique) |
| Dec. 7, 2014 | Task 3 Age & Gender Distribution of Cities in Netherlands CLICK Choropleth Map, Bar and Pie Charts (Region-based Technique) |

Figure 1. Index page

The remaining parts of this report are organized as follow: In section 2, the D3 library will be introduced while section 3 introduces the dataset. Each task is elaborated respectively in section 4 - 6, in the order of its task description, motivation, design and implementation. Section 7 shows the tests of the techniques and observations, pros and cons of each task. Section 8 concludes the report.

2. Tool: D3

D3 is a JavaScript library for manipulating documents based on data. D3 helps people bring data to life using Hyper Text Markup Language (HTML), Scalable Vector Graphics (SVG) and Cascading Style Sheets (CSS). D3 allows binding arbitrary data to a Document Object Model (DOM), and then applying data-driven transformations to the document. For example, people can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction. With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviors for interaction and animation. D3 is a very resourceful library with a large number of default examples online. D3's functional style allows code reuse through a diverse collection of components and plugins. We use the latest version (3.5.0) and every time we inline the source script from <http://d3js.org/d3.v3.min.js> to local HTML to keep it simple and straightforward.

3. Data Set

The dataset originates from Open CBS [3] and it consists of shape information of Dutch municipalities in GEOJson format (cities-geometry.json), and a tab-separated file containing many statistics of these municipalities (cities-data.txt). The data is updated in 2014, but it is based on the CBS District and Neighborhood Map 2012 [4]. The data set contains 58 various categories of data for totally 415 municipalities in the Netherlands, such data includes general information of each city (e.g., name, area, population, density of population, etc), and specific information in terms of age, foreigners, marital status, vehicles, households, etc. The GEOJson file contains geographical information of each city so it is helpful to generate a map for all cities, but it does not contain all information in the .txt file, which means we have to rely on .txt file mostly. The original data in the .txt file is sorted by the code of municipalities. In order to implement several techniques, we have to transfer the cities-data.txt into .tsv extension for further usage. Furthermore, the last row in the .tsv file is useless so we just delete it. For our three tasks, this data set is totally enough, so we did not add other extra data. The statistics and description related to the dataset provided can be found in Appendix.

4. Task 1: Foreigners Proportion of Cities in the Netherlands

In this section, we solve the first task, which is to visualize the foreigners' proportion of cities in the Netherlands.

4.1. Task

This task is inspired by the Bureau of Statistics of the Netherlands and the Bureau of Immigration of the Netherlands. In the case that the latter wants to know foreigner proportion of each city in the Netherlands in a very interactive way, so they ask the former to provide such data and then we help them to create such a graph by the tool of D3. They might want to have following requirements:

- (1) Sort the data from the perspective of foreigners, i.e., users can select which foreigners they want to investigate;
- (2) Show every city's data in one graph so that they can make comparison among them directly;
- (3) Although there are data for more than 400 cities, do not overlap these data as much as possible;
- (4) Provide relevant information for each city, e.g., the number of inhabitants, density of population, etc.;
- (5) Classify those cities into several categories so that it is easier to distinguish them.

4.2. Motivation

According to the aforementioned task requirements, we found that scatterplot could be a good way for such a task, since usually this point-based technique is able to contain large-scale data easily. Given the example [5-7], this technique is suitable for the data of more than 400 cities, which hits requirement 2. What's more, the x-axis, y-axis, color of scatter points, and even the size of points can represent different data respectively, which satisfies their requirement 4. One of these attributes, the color of points, can simply define the category of those cities, so it accords to requirement 5. For the first requirement, we can easily give a drop-down menu for users to select the foreigner group they want to investigate, and generate one scatterplot for each group. Inspired by [8], requirement 3 can be satisfied by zoom-in / zoom-out feature among 400 points to avoid overlapping.

4.3. Technical Design

The design is based on the requirements and motivations, using the D3. The design can be found in figure 2. Users can choose the foreigners' group in the listed drop-down menu, which contains 7 categories of foreigners. When choosing one from the list, the scatterplot will generate the foreigners proportion in every Dutch city. In the scatterplot, each bubble represents one city; the size of the bubble represents the density of inhabitants (the value has been square-rooted), which means how many people live in one squared kilometers. The colors of the bubbles are the classifications of urban character, as shown in the legend, which classifies all municipalities into 5 categories by their average number of addresses per squared kilometers. The x-axis represents the number of inhabitants in this city (pow-scale), and the y-axis is the percentage of the specific foreigners you choose.

When users put mouse on one of the bubbles, the city's name, number of inhabitants and density of inhabitants will not only respectively show on the text box attached to it, but also shows on the right side of the webpage (just below the drop-down menu). What is more, the bubble will automatically be enlarged to its double size and two lines are generated, pointing to the corresponding values on x-axis and y-axis. One more interesting interaction: you can zoom in or zoom out to see more details of the bubbles by mouse wheeling or double-click on the bubbles.

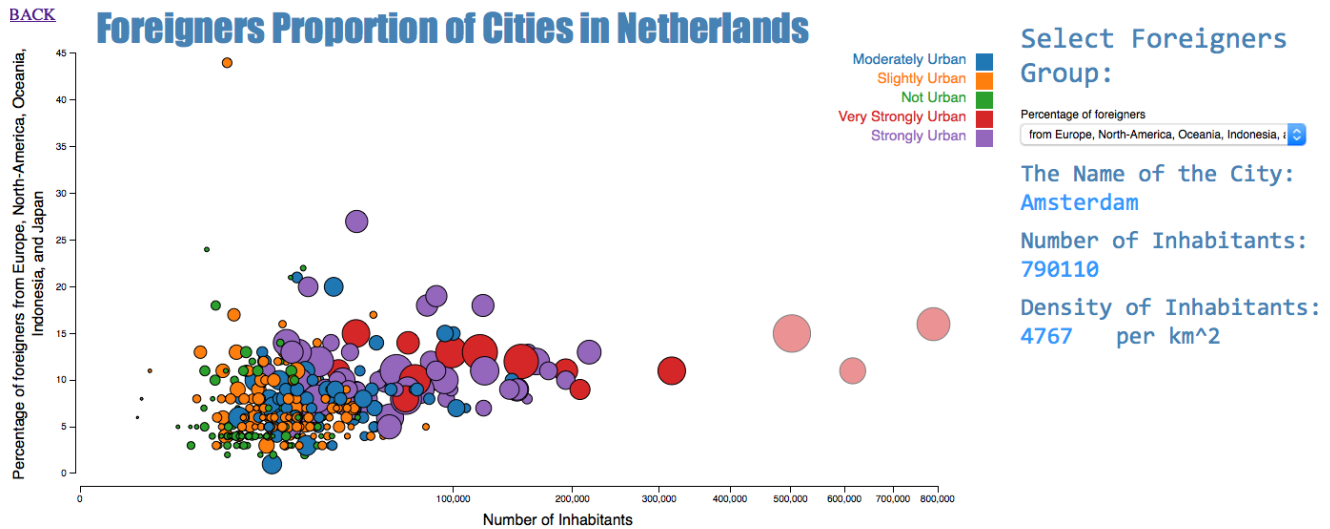


Figure 2. Interactive Scatterplot for Task 1

4.4. Implementation

In this part, every part of this application in terms of code will be explained in detail.

A. Drop-down menu

The drop-down menu is written directly in HTML code, and contains 7 options in the list (see figure 3). When clicking on one of them, the function `alertChange()` is triggered, the value of selection will be parsed and the whole code for generation of the scatterplot is written inside this function, so a corresponding scatterplot will be shown.

The function `alertChange()` is binded with the drop-down menu by the code:

```
d3.select("#item1").on("change", alertChange);
```

The parsing from selection to `selectedValue` is on this code:

```
var selectedValue = parseInt(d3.event.target.value);
```

Translating the `selectedValue` into the generation of a scatterplot is by a 'switch-case', since we are not able to select one of the city's value from the data set directly by the value, so that we have to use this 'switch-case', and it increases the length of the code though. Everything needed to be changed is written in the 'switch-case', the rest is defined outside the function `alertChange()`.

Select Foreigners Group:

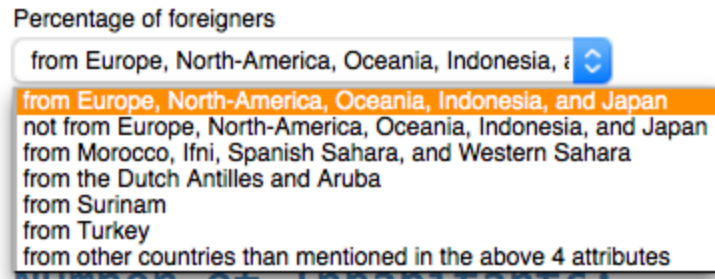


Figure 3. Drop-down menu

B. Data Input

The data is input through the following code provided by D3, in which the ‘dataset’ is a global variable for other functions to further use:

```
d3.tsv("cities-data.tsv", function(error, data) {
    dataset = data;
    ..... }
```

In the above function(error, data) {}, the initial scatterplot is generated, with P_WEST_AL as y-axis.

C. Scatterplot

As said previously, the scatterplot first is initialized by P_WEST_AL as y-axis in the code of data input. And then each time users select one option from the drop-down menu, the y-axis and all the bubbles will be correspondingly changed.

- 1) X-Axis: The x-axis is scale by the power of exponent 0.4, for the concern of too many cities are crowded in the range of [0, 100,000]. So scaling by power can to some extent decrease this crowd. The code is as follow:

```
d3.scale.pow().exponent(.4).range([0, 900]);
```

A table of “Number of Inhabitants” is attached below.

- 2) Y-Axis: The y-axis is changed according to users’ option in the drop-down menu, and it is always scaled linearly. The label attached is changed as well.
- 3) Size of bubbles: The size of bubble is also changed every time by users’ option, so it is also written into the aforementioned ‘switch-case’. The value has been square-rooted already because if using the value of BEV_DICHTH, some radius of bubbles will be too large to cover others and some others will be too small to find it. So we use square-root to pre-process this data:

```
circle.data(dataset)
    .enter().append("circle")
    .attr("class", "dot")
    .attr("r", function(d) { return Math.sqrt(d.BEV_DICHTH) / 4; })
```

- 4) Color of bubbles: Color of bubbles shows the classification of the selected city, and the color is defined by:

```
var color = d3.scale.category10();
```

And how it uses the data is as follow:

```
circle.data(dataset)
  .enter().append("circle")
  .attr("class", "dot")
  .style("fill", function(d) { return color(d.STED); })
```

D. Legend

It is important that users can look up what the colors mean, so we add a legend on the top-right corner of the graph (see figure 4). The colors correspond with the color of bubbles, representing the classification of a selected city. From the perspective of code, it uses one 'g' of the 'legend' class on the 'svg', adding rectangles and texts with it.



Figure 4. Legend

E. Interaction

Interaction increases fun and also provides more detailed information on the graph.

1) Mouse On / Mouse Off:

Each time when generating a scatterplot, one function `mouseOn()` and one function `mouseOff()` are deployed on bubbles:

```
circle.data(dataset)
  .enter().append("circle")
  .attr("class", "dot")
  .on("mouseover", mouseOn)
  .on("mouseout", mouseOff)
```

In the function `mouseOn()`, it implements doubling the size of bubbles, and adding one horizon and one vertical line to point the value on the axes (see figure 5). Also, showing a text attached to this bubble, showing value of the x-axis, y-axis, size of bubble and city name.

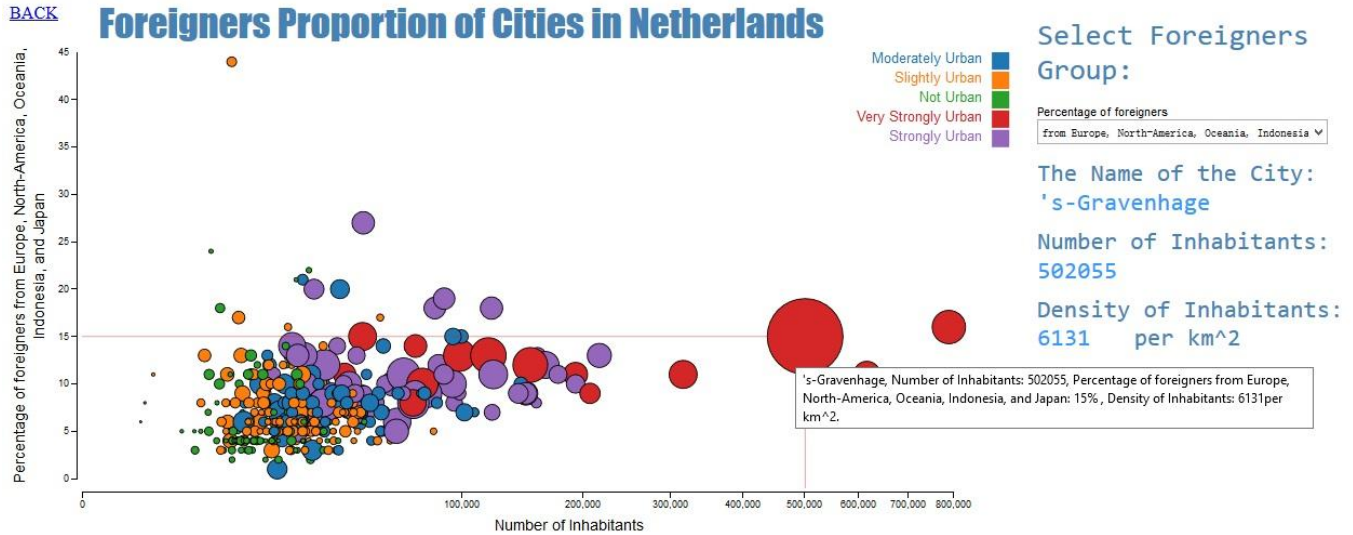


Figure 5. Effect of Mouse-On

The function `mouseOff` have the effect of fade, which means putting mouse away the bubble, the bubble will be faded by change its 'opacity' style into 0.5.

2) Zoom-In / Zoom-Out:

Recalling requirement 3: to avoid overlapping as much as possible, we have to add the effect of zoom in and zoom out, so that people can see the bubbles in a more diverted way, which means the bubbles will not be that crowded after zoom-in (see figure 6). When zoom-in/zoom-out, people have to put mouse on one bubble and then mousewheel or double-click it, the whole scatter will zoom-in/zoom-out with the center of this bubble. To implement such a zoom-in/zoom-out interaction, we add such code right after generating the scatterplot:

```
svg.call(d3.behavior.zoom().x(x).y(y).on("zoom", zoom1));
```

Besides, define the call-back function `zoom()` for each case:

```
function zoom1() {
  svg.selectAll(".dot")
    .attr("cx", function(d) { return x(d.AANT_INW); })
    .attr("cy", function(d) { return y(d.P_WEST_AL); });
  d3.select('.x.axis').call(xAxis);
  d3.select('.y.axis').call(yAxis);}

```

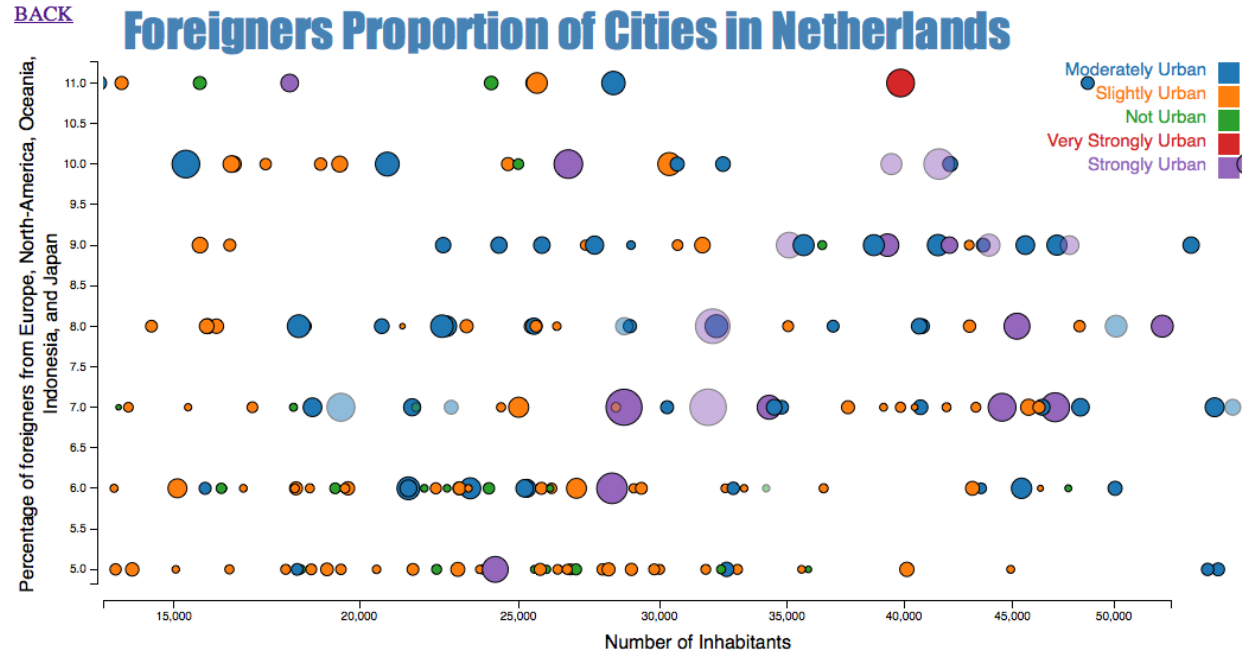
[BACK](#)

Figure 6. Effect of Zoom-In

3) Drag:

With the bonus of zoom-in/zoom-out, users can also drag one of the bubbles and move the whole graph into any position.

5. Task 2: General Marital Status in the Netherlands

In this section, we solve the second task, which is to visualize the general marital status in the Netherlands.

5.1. Task

Assume that the Bureau of Civil Affairs of Netherlands wants to show an interactive graph on its website, regarding the general marital status in the Netherlands. To visualize general marital status in one graph is not an easy job, since such status includes not only unmarried, married, but also divorced and widows. To connect the relationships of each specific status, and show the trend, they proposed such requirements:

- (1) Show trends of every specific marital status for all cities in the Netherlands;
- (2) The graph should be easy to find outliers, correlation and extremes;
- (3) Make the graph as scalable as possible.

5.2. Motivation

Based on the above requirements, parallel coordinate plots can easily fulfill this task. Just like [9-11], the trends, outliers, correlation and extremes can be seen from the clusters of lines between two dimensions. What's more, all the axes are draggable, which means they can reorder the dimensions so that users could easily see different comparisons, therefore it satisfies requirement 1 and 2. Requirement 3 could also be achieved since parallel coordinate plots can handle large-scale data.

5.3. Technical Design

The technical design is based on the requirements and motivations, using D3. The design of this parallel coordinate plot can be found in figure 7. Users can directly see all the specific marital status, which is represented by each axis. Each axis is represented by one color, and a legend on the right side of the page shows its meaning. The line clusters between two axes are showing the trends, outliers, correlation and extremes, giving the general information of marital status in the Netherlands. The interaction is easy, just to drag the name of the axis, and move it to anywhere you want, so it provides a simple way to see comparisons between different parameters.

[BACK](#)

General Marital Status in Netherlands

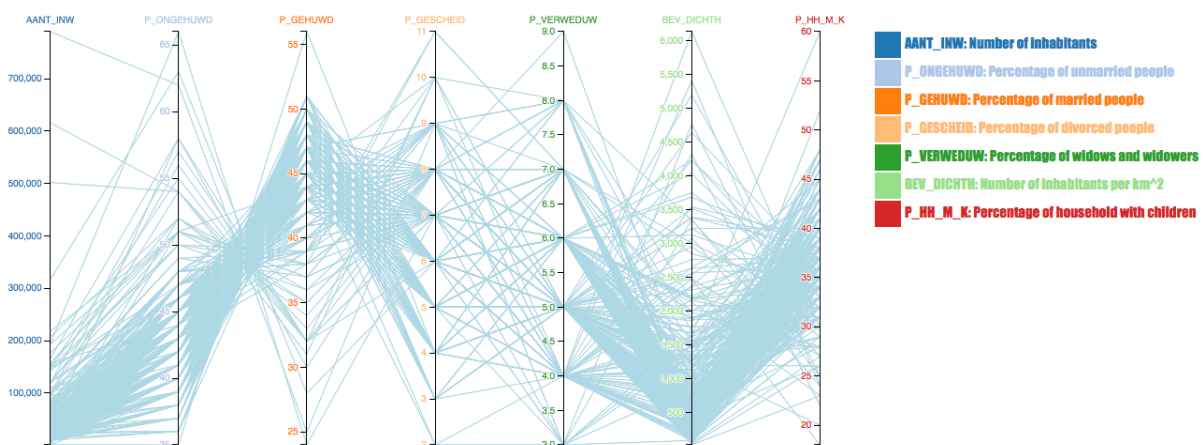


Figure 7. Draggable Parallel Coordinate Plot for Task 2

5.4. Implementation

In this part, every part of this application in terms of code will be explained in detail.

A. Axes

Each axis is drawn on one 'g' on the 'svg'. All axes are scaled linearly, showing corresponding data on them, like number of inhabitants, percentage of married/unmarried/divorced/widows, number of inhabitants per km² and percentage of household with children.

B. Data Input

The data input is also using .tsv file as the previous task:

```
d3.tsv("cities-data.tsv", function(cities) {
    .....
})
```

But the x domain changed according to the dimension, filtered to select the 7 values we want to show in the graph.

C. Lines

We added grey background lines for context and blue foreground lines for focus. For example, we use such code for the foreground lines:

```
foreground = svg.append("g")
    .attr("class", "foreground")
    .selectAll("path").data(cities).enter()
    .append("path").attr("d", path);
```

D. Legend

The implementation of legend is the same as the previous legend in Task 1, and here we add 7 colored categories for each dimension.

E. Interaction

In order to implement the draggable interaction (see figure 8), we have to call the behavior of 'dragstart', 'drag' and 'dragend' for the class of 'dimension', like the code here:

```
var g = svg.selectAll(".dimension")
    .data(dimensions).enter().append("g")
    .attr("class", "dimension")
    .call(d3.behavior.drag())
    .origin(function(d) { return {x: x(d)}; })
    .on("dragstart", function(d) {
        dragging[d] = x(d);
        background.attr("visibility", "hidden");
    })
    .on("drag", function(d) {
        dragging[d] = Math.min(width, Math.max(0, d3.event.x));
```

```

.....
})
.on("dragend", function(d) {
  delete dragging[d];
  .....
}));

```

[BACK](#)

General Marital Status in Netherlands

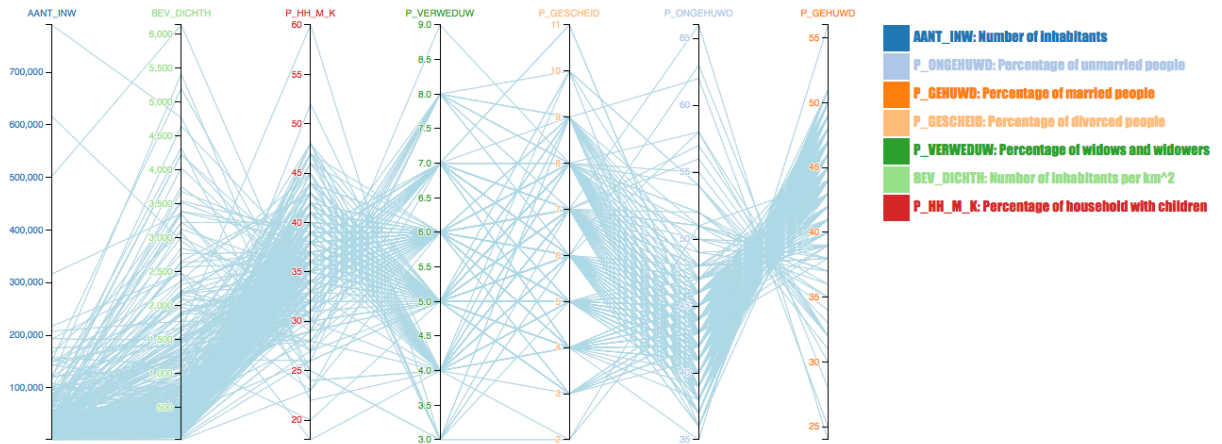


Figure 8. The Parallel Coordinate Plot after Dragging

6. Task 3: Age & Gender Distribution of Cities in the Netherlands

In this section, we solve the third task, which is to visualize both the age and gender distribution of cities in the Netherlands.

6.1. Task

Assume that one cloth manufacturer company that manufacture different clothes for different age groups and genders is interested to visualize the number of inhabitants of all the cities in the Netherlands. The graph should also contain information of percentage of different age groups and genders in each city, since they can use such data to investigate the market size of some certain clothes target at certain ages for each city and the company would be convenient to distribute their warehouse of different clothes in different cities. They might want to have the following requirements:

- (1) All cities would be easy to visualize from the perspective of geography;
- (2) Each city should provide insights on different age groups;
- (3) Each city should also provide insights on percentage of men and women;
- (4) Provide other related information for cities.

6.2. Motivation

According to the aforementioned task requirements, we found that choropleth map could be a good way for such a task, since it helps to visualize how a variable (number of inhabitants) varies across a geographic area, which can basically satisfy requirement 1. Inspired from different examples [12-15], we opted to use choropleth map to show municipalities in the Netherlands with color representing number of inhabitants, with other related information showing on attached texts, it is easy to fulfill requirement 4. A static choropleth map is simple and provides limited information, but a choropleth map with interaction helps to explore and visualize different variables (percentage of different age groups, percentage of men and women) associated with a dataset which fulfills the above mentioned task requirements. We choose bar chart to interactively show different age groups since we've got 19 age groups from the dataset, the bar chart can show it clearly. Besides, we use donut chart to visualize percentage of men and women in a very crisp way. They hit the requirement 2 and 3 respectively.

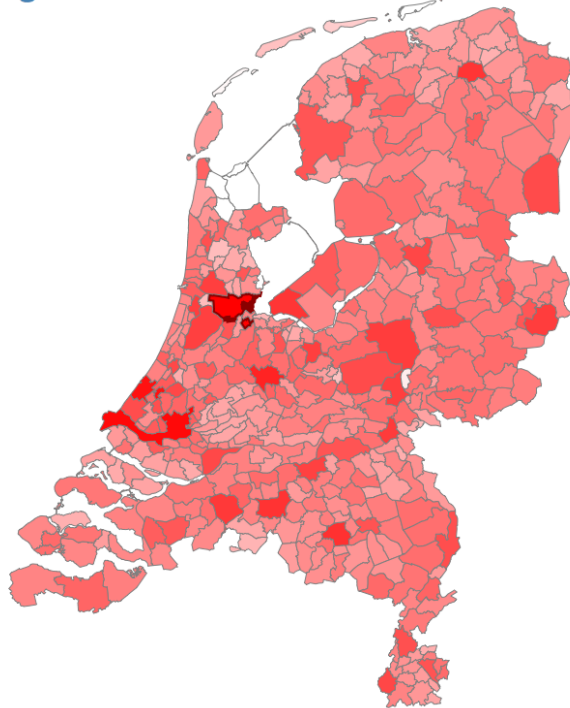
6.3. Technical Design

This page includes choropleth map with bar chart and donut chart (see figure 9). On the choropleth map, it shows all the cities in the Netherlands, with the color representing the number of inhabitants of the corresponding city, which is scaled by log from white (minimum) to red (maximum) with single hue progression. When you put mouse on one city, it will interactively generate a corresponding bar chart for percentage of different age groups, and generate a donut chart for genders for this selected city.

If you click a city on the map, the map will zoom-in and click again on it will zoom-out back to the original map. Mouse on a city can show its border, and it shows the information of city name, area, and number of inhabitants not only on the text attached to city on the map but also on the texts on the top-right corner of the page.

[BACK](#)

Age & Gender Distribution of Cities in Netherlands



Name of the Selected City: **Amsterdam**
 Total Area of This City: **21932** hectares
 Number of Inhabitants: **790110**

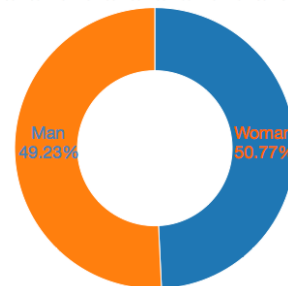
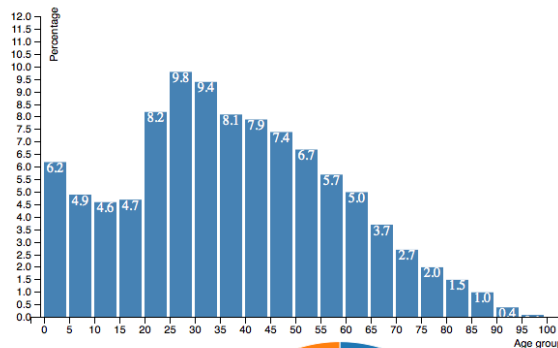


Figure 9. Combination of Choropleth Map, Bar Chart and Donut Chart for Task 3

6.4. Implementation

In this section, every part of this application in terms of code will be explained in detail.

A. Choropleth Map

To draw the choropleth map, path data is acquired from JSON file to draw the boundaries of the cities of the Netherlands. JSON file provides latitude and longitude to draw these boundaries. It is important to note that longitude is always listed first despite representing as traditional lat/lon.

Path: It translates the coordinates into SVG path codes. The following section of the code helps to achieve this:

```
var path = d3.geo.path();
```

Projections: It converts 3D space on to 2D plane. D3 has built-in projections. The following section of the code help to achieve this:

```
var projection = d3.geo.albers();
```

Color Scale: We scale the color by log that is because in the case of linear scale, most of the cities will be white while some big cities are totally red. By using log scale, the problem is solved, and people can see more colors instead of just maximum (white) and minimum (red). The definition is here:

```
var logColorScale = d3.scale.log().domain([0.0, 100.0]).range(["white", "red"]);
```

B. Data Input

Data is read by using queue, which is a JavaScript asynchronous helper library. The following section of the code read the data.

```
queue()
```

```

.defer(d3.json, "cities-geometry.json")
.defer(d3.tsv, "cities-data.txt", function(d) { if(d.Naam != "") {
    cityData.set(d.Code, +d.AANT_INW); } })
.defer(d3.tsv, "cities-data.txt", function(d) { cityDataAll.set(d.Code, d); })
.await(dataLoaded);

```

queue() - initializes Queue.js and loading queue.

defer() - used for each file to allow them to load separately.

await() - it runs a function once the data are loaded.

C. Bar Chart

Bar chart interactively shows the age groups of a selected city.

X-Axis: Represent the different age groups.

Y-Axis: Represent percentage.

Rectangle is used to draw the bars, and texts with information of percentage for each groups are attached on top of the bars.

D. Donut Chart

Donut chart displays percentage of men and women of the selected city. The following section of the code helps to achieve that.

```

svg.selectAll(".arc")
  .data(donut(manwoman))
  .enter().append("g")
  .attr("class", "arc")
  .append("path")
  .attr("d", arc)
  .style("fill", function(d, i) { return color(i); })

```

E. Map Interaction

- 1) MouseEnter/ MouseOut: The border of the selected city is highlighted when user put mouse on a city. The following section of the code achieves the desired result.

```

.on("mouseenter", function(d,i) {
    d3.select(this).transition().duration(300)
    .style({'stroke-opacity':1,'stroke':"darkred", 'stroke-width': "5px"});
})
.on("mouseout", function(d,i) {
    d3.select(this).transition().duration(300)
    .style({'stroke-width':"1px",'stroke':"grey"});
})

```

- 2) Click: Choropleth map also supports zoom-in/zoom-out feature (see effect of zoom-in on figure 11). Although it is not required to meet the requirement but it makes the map more interactive and helps to focus on one city. To implement such a zoom-in/zoom-out interaction, we added an event on click:

```

function clicked(d) {
    if (active.node() === this) return reset();
}

```

```

    active.classed("active", false);
    active = d3.select(this).classed("active", true);

```

```

    .....

```

```

}

```

For zoom-out we add the following code:

```

function reset() {
    active.classed("active", false);
    active = d3.select(null);
    .....
}

```

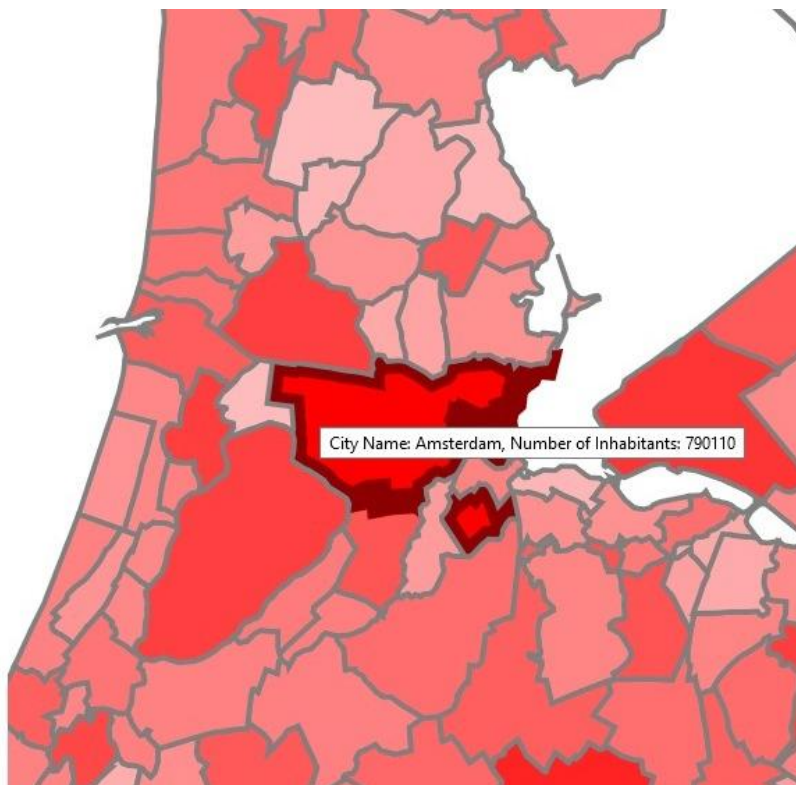


Figure 10. Effect of Zoom-In when Clicking on Amsterdam

F. Bounce Interaction of Bar Chart

The following section of the code implements the bounce interaction of bar chart.

```

...//previous states of bars
.transition()
.delay(function(d,i){ return 100 * i;})
.ease("bounce")
.duration(1500)
...//states of bars after bounce

```

transition() - It handles the interpolation.

ease() - It controls the quality of motion.

delay() - It specifies when the transition begins.

duration() - It defines the duration for the whole interaction.

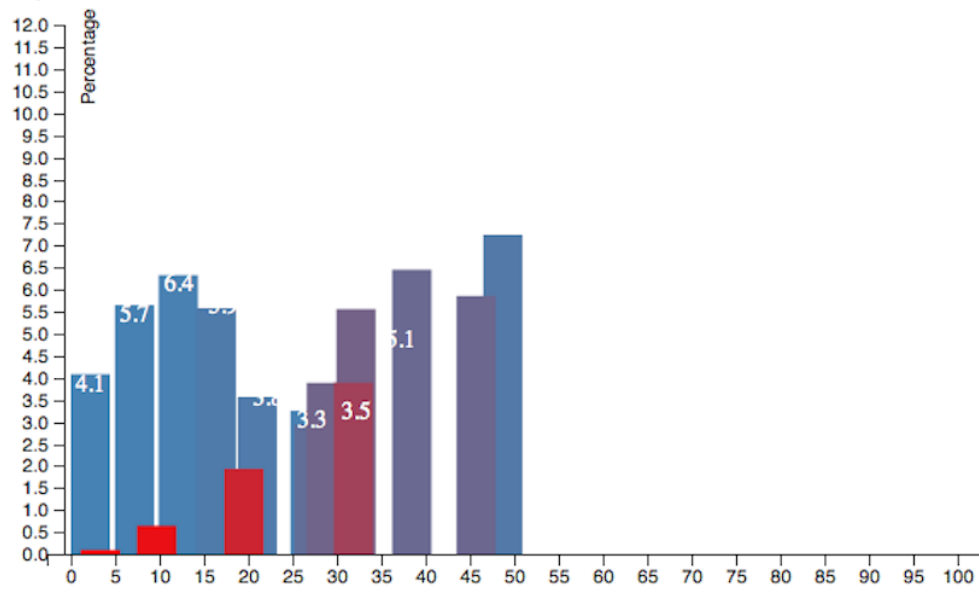


Figure 11. One frame in the Bounce Interaction of the Bar Chart






7. Tests

In this section, we are going to test each task on different browsers, and also we will tell the pros and cons using this technique and show the basic results and interesting observations from the analysis of information visualization.

7.1. Browsers and Environment

In the table I, we tested all the 5 browsers on a Windows laptop, with the resolution of 1366 * 768. And we opened each task in a maximized window. All browsers are latest versions, downloaded from their official websites. All tasks need to run when accessing to the Internet, since it needs to inline the D3 library online. We installed XAMPP web server to test the application for different web browsers mentioned in the table I. Test was done by with and without a (local) web server respectively. Chrome, IE and Opera do not allow loading data from a local file on the disk, so it does not work without a web server. Other cases can perfectly show the three visualized graphs on the window as what we expected.

Table I. Browsers support

| Browsers | Chrome  | IE  | Firefox  | Safari  | Opera  |
|----------------------|--|--|---|--|---|
| Without a web server | X | X | √ | √ | X |
| With web server | √ | √ | √ | √ | √ |

7.2. Task 1

A. Observations: We can tell from the bubble distribution according to colors and x-axis on the graph that more urban cities usually have more inhabitants. Furthermore, inspired from the bubble distribution according to colors and bubble sizes, we can find that bigger type of cities enjoys more density of population, and also more people one city has, usually more density it is.

When considering about the foreigners, bigger urban usually have more percentage of non-western people, while smaller cities have less non-western people. The western people in Netherlands occupied more than non-western foreigners. Among all kinds of foreigners, the category “from other countries not mentioned on the above 4 categories” has the most percentage, while the “from the Dutch Antilles and Aruba” is the least. Also, the density of population does not have a relationship with percentage of foreigners, but it correlates with the city type, i.e., bigger urban has more density of population, which makes sense.

B. Pros: It is available to display four kinds of data in one diagram; The pow-scale of x-axis and zoom-in/zoom-out could avoid overlapping of bubbles; The different area of bubbles filters unessential data and gives an emphasis.

C. Cons: The data is not quite readable, which means we cannot reach conclusions directly from the diagram.

7.3. Task 2

A. Observations: This diagram mainly gives users some ideas by dragging the axes. After dragging and comparing different value, we can simply see that most cities have less than 200,000 inhabitants. Most cities live 40% - 50% of married people, and 40% - 50% of unmarried people. This fact is directly got from the data, though it does not make sense since their addition are not 100%. If put AANT_INW and P_GEHUWD together, we can find that cities with more inhabitants have less percentage of married people, which indicates that people in big cities are harder to get married compared with small cities. This can also be verified when putting AANT_INW and P_GESCHIED together, usually cities with more inhabitants have more divorced percentage. Putting P_VERWEDUW and BEV_DICHTH together, we can tell from the cluster of lines that the more density of population one city has, it might have less widows or widowers, which also makes sense since in denser cities people have more chances to meet new people.

B. Pros: This diagram gives a clue for trends, outliers, correlation and extremes, just like the fact that we list in the observations; the information it can provide is high-level, so we can see the general marital status of a whole country.

C. Cons: It is unavailable to select one specific city to see its detailed data.

7.4. Task 3

A. Observations: Naturally, the colors on the map tell that big-area cities are not necessarily have more inhabitants. But when know the geographic locations of some big cities such as Amsterdam, Den Haag, we can see that the colors are more red than others, and this makes sense.

The age distribution leans to the age group 5 - 20 years old and 40 - 70 years old, which means most cities lack laborers who are at their best working ages (20 - 40 years old). But for bigger cities, the case is different. Big cities like Amsterdam and Eindhoven, they enjoy large percentage among the age group 20 – 45 years old, which explained why the previous phenomenon happens, because most laborers go to bigger cities for work. The gender distribution chart shows that genders in most cities are even.

B. Pros: The choropleth map gives a straightforward view when selecting a specific city; The interaction is so fancy that can attract users' attention; Information given on the map and two charts are plentiful, and the value are readable; Bar chart and donut chart gives users a convenience to easily make comparison.

C. Cons: It does not classify the cities into provinces.

8. Conclusions

We implemented three different visualization techniques based on D3, HTML and CSS to solve three specific tasks according to the requirements of each task. Each visualization technique helps users to visualize the dataset easily and provides some interesting insights interactively which further helps to analyze the dataset associated with task set. All three techniques are tested with a local web server and without a webserver on all modern web browsers. Some interesting observations are analyzed, and pros and cons are concluded afterwards. We find that the information visualization really helps to understand the data easier. Finally, what we learned from this project is also applicable for the future research tasks.

Reference

- [1] Wikipedia. *Information visualization*. Published on the WWW at http://en.wikipedia.org/wiki/Information_visualization, November 25, 2014.
- [2] Mike Bostock. *Data-Driven Documents*. Published on the WWW at <http://d3js.org/>, December 7, 2014.
- [3] Beatriz del Saz, Thijs Candel and Steven. *CBS Open Data*. Published on the WWW at <http://www.opencbs.nl/>, December 7, 2014.
- [4] Central Bureau of Statistics. *District and neighborhood map 2012*. Published on the WWW at <http://www.cbs.nl/nl-NL/menu/themas/dossiers/nederland-regionaal/publicaties/geografische-data/archief/2013/2013-2012-b68-pub.htm>, December 7, 2014.
- [5] Mbostock. *Scatterplot*. Published on the WWW at <http://bl.ocks.org/mbostock/3887118>, October 14, 2012.
- [6] Nsonnad. *Scatterplot: Social trust vs ease of doing business*. Published on the WWW at <http://bl.ocks.org/nsonnad/4481531>, January 8, 2013.
- [7] Daniel Resende, Rafael Abensur. *Como ganhar a Copa? A partir das estatísticas e da economia*. Published on the WWW at <https://www.verios.com.br/blog/como-ganhar-a-copa/#x-axis-menu>, June 4, 2014.
- [8] Mbostock. *Drag + Zoom*. Published on the WWW at <http://bl.ocks.org/mbostock/6123708>, July 31, 2013.
- [9] *Parallel Coordinates*. Published on the WWW at <https://syntagmatic.github.io/parallel-coordinates/>, December 7, 2014.
- [10] Jasondavies. *Parallel Coordinates*. Published on the WWW at <http://bl.ocks.org/jasondavies/1341281>, November 5, 2011.
- [11] *Nutrient Contents - Parallel Coordinates Opacity: 20%*. Published on the WWW at <http://exposedata.com/parallel/>, December 7, 2014.
- [12] Mbostock. *Choropleth*. Published on the WWW at <http://bl.ocks.org/mbostock/4060606>, November 12, 2012.
- [13] Mbostock. *Bar Chart*. Published on the WWW at <http://bl.ocks.org/mbostock/3885304>, October 13, 2012.
- [14] Mbostock. *Click-to-zoom via transform*. Published on the WWW at <http://bl.ocks.org/mbostock/3887193>, March 26, 2012.
- [15] Mbostock. *Donut Chart*. Published on the WWW at <http://bl.ocks.org/mbostock/2206590>, October 14, 2012.

APPENDIX

GM CODE [string]

This code provides a numerical identity to a municipality.

GM NAAM [string]

The official name of the municipality.

WATER []

Has no relevance here.

OAD [number]

Average number of addresses per square kilometer.

STED [number]

Describes the urban character using the following classification:

1. very strongly urban ≥ 2500 addresses per km²
2. strongly urban 1500 - 2500 addresses per km²
3. moderately urban 1000 - 1500 addresses per km²
4. slightly urban 500 - 1000 addresses per km²
5. not urban < 500 addresses per km²

AANT INW [number]

Number of inhabitants.

AANT MAN [number]

Number of men.

AANT VROUW [number]

Number of women.

P 00 14 JR [percentage]

Percentage of inhabitants aged 0 to 15 years.

P 15 24 JR [percentage]

Percentage of inhabitants aged 15 to 25 years.

P 25 44 JR [percentage]

Percentage of inhabitants aged 25 to 45 years.

P 45 64 JR [percentage]

Percentage of inhabitants aged 45 to 65 years.

P 65 EO JR [percentage]

Percentage of inhabitants aged 65 years and older.

P ONGEHUWD [percentage]

Percentage of unmarried people.

P GEHUWD [percentage]

Percentage of married people.

P GESCHEID [percentage]

Percentage of divorced people.

P VERWEDUW [percentage]

Percentage of widows and widowers.

BEV DICHTH [number]

Number of inhabitants per km².

AANTAL HH [number]

Number of households.

P EENP HH [percentage]

Percentage of single households.

P HH Z K [percentage]

Percentage of households without children.

P HH M K [percentage]

Percentage of households with children.

GEM HH GR [number]

Average number of people in all households.

P WEST AL [percentage]

Percentage of foreigners from Europe, North-America, Oceania, Indonesia, and Japan.

P N W AL [percentage]

Percentage of foreigners not from Europe, North-America, Oceania, Indonesia, and Japan.

P MAROKKO [percentage]

Percentage of foreigners from Morocco, Ifni, Spanish Sahara, and Western Sahara.

P ANT ARU [percentage]

Percentage of foreigners from the Dutch Antilles and Aruba.

P SURINAM [percentage]

Percentage of foreigners from Surinam.

P TURKIJE [percentage]

Percentage of foreigners from Turkey.

P OVER NW [percentage]

Percentage of foreigners from other countries than mentioned in the above 4 attributes.

AUTO TOT [number]

Number of cars.

AUTO HH [number]

Number of cars per household.

AUTO LAND [number]

Number of cars per km².

BEDR AUTO [number]

Number of company cars (minivans, trucks, etc.).

MOTOR 2W [number]

Number of motorcycles, including scooters.

OPP TOT [number]

Total land and water area in hectares.

OPP LAND [number]

Land area in hectares.

OPP WATER [number]

Water area in hectares.

P 00 04 JR [percentage]

Percentage of inhabitants aged 0 to 4 years.

P 05 09 JR [percentage]

Percentage of inhabitants aged 5 to 9 years.

P 10 14 JR [percentage]

Percentage of inhabitants aged 10 to 14 years.

P 15 19 JR [percentage]

Percentage of inhabitants aged 15 to 19 years.

P 20 24 JR [percentage]

Percentage of inhabitants aged 20 to 24 years.

P 25 29 JR [percentage]

Percentage of inhabitants aged 25 to 29 years.

P 30 34 JR [percentage]

Percentage of inhabitants aged 30 to 34 years.

P 35 39 JR [percentage]

Percentage of inhabitants aged 35 to 39 years.

P 40 44 JR [percentage]

Percentage of inhabitants aged 40 to 44 years.

P 45 49 JR [percentage]

Percentage of inhabitants aged 45 to 49 years.

P 50 54 JR [percentage]

Percentage of inhabitants aged 50 to 54 years.

P 55 59 JR [percentage]

Percentage of inhabitants aged 55 to 59 years.

P 60 64 JR [percentage]

Percentage of inhabitants aged 60 to 64 years.

P 65 69 JR [percentage]

Percentage of inhabitants aged 65 to 69 years.

P 70 74 JR [percentage]

Percentage of inhabitants aged 70 to 74 years.

P 75 74 JR [percentage]

Percentage of inhabitants aged 75 to 79 years.

P 80 84 JR [percentage]

Percentage of inhabitants aged 80 to 84 years.

P 85 89 JR [percentage]

Percentage of inhabitants aged 85 to 89 years.

P 90 94 JR [percentage]

Percentage of inhabitants aged 90 to 94 years.

P 95 EO JR [percentage]

Percentage of inhabitants aged older than 95 years.