Kosovo Mosaic Visualization Documentation

|  |  |
| --- | --- |
| **Document** | Kosovo Mosaic Visualization Documentation |
| **Version** | 0.1 |
| **Last Updated** | 22 December 2015 |
| **Updated by** | Brett Romero |

# Introduction

The Kosovo Mosaic visualization is a tool designed to help users understand and explore data collected as part of the Kosovo Mosaic project. This data contains a variety of data points reflecting the level of satisfaction or dissatisfaction of citizens for a range of public services and processes. The data also covers all municipalities and 5 (non-consecutive years) – all of which can be viewed.

The visualization is designed to provide access to this data in a user-friendly way while also allowing even basic users to drill down to the detailed statistics relevant to their municipality.

# Data

## Indicators

The data used in the visualization is primarily based around a set of 36 indicators (see Table 1). These indicators reflect a range of public services, processes and areas of public responsibility. The values reflected in the dataset represent (based on the users selections) either the percentage of citizens (dis)satisfied or very (dis)satisfied with the particular service/process/area of responsibility. The satisfied and dissatisfied percentages are the inverse of each other, with the difference resulting from the people who responded ‘Don’t know”.

Please note that not all indicators were used in all years.

#### Table 1 – 36 Indicators

|  |  |
| --- | --- |
| 1 | Access to and quality of preschool education |
| 2 | Cemetery maintenance |
| 3 | Cultural activities |
| 4 | Electricity supply |
| 5 | Environmental protection |
| 6 | Family medical centers |
| 7 | Fire and emergency services |
| 8 | Hospitals |
| 9 | Local road maintenance |
| 10 | Maintenance of inter-municipal roads and highways |
| 11 | Management of municipal funds |
| 12 | Municipal administration |
| 13 | Municipal assembly |
| 14 | Municipal parks and squares |
| 15 | Nature and species conservation |
| 16 | Phone and postal services |
| 17 | Primary and secondary schools/education |
| 18 | Procedures for tax payment |
| 19 | Protection of cultural heritage |
| 20 | Public health |
| 21 | Public lighting |
| 22 | Public procurement/tenders |
| 23 | Public transport |
| 24 | Recruitment of local staff |
| 25 | Representation of ethnic minorities |
| 26 | Representation of women |
| 27 | Sewage and sanitation |
| 28 | Sidewalks |
| 29 | Social services |
| 30 | Sports activities |
| 31 | The mayor |
| 32 | Traffic and parking control and regulation |
| 33 | Urban and rural planning |
| 34 | Waste collection services |
| 35 | Water supply |
| 36 | Youth activities |

## Main Problems

In addition to the indicator data, data on what citizens perceived to be the major problems in their municipality is also shown in the visualization. The data reflects a survey question that asked respondents what they perceived to be the bigger problems in their municipality. These results were then aggregated and the data shows the 3 most commonly identified problems in each municipality.

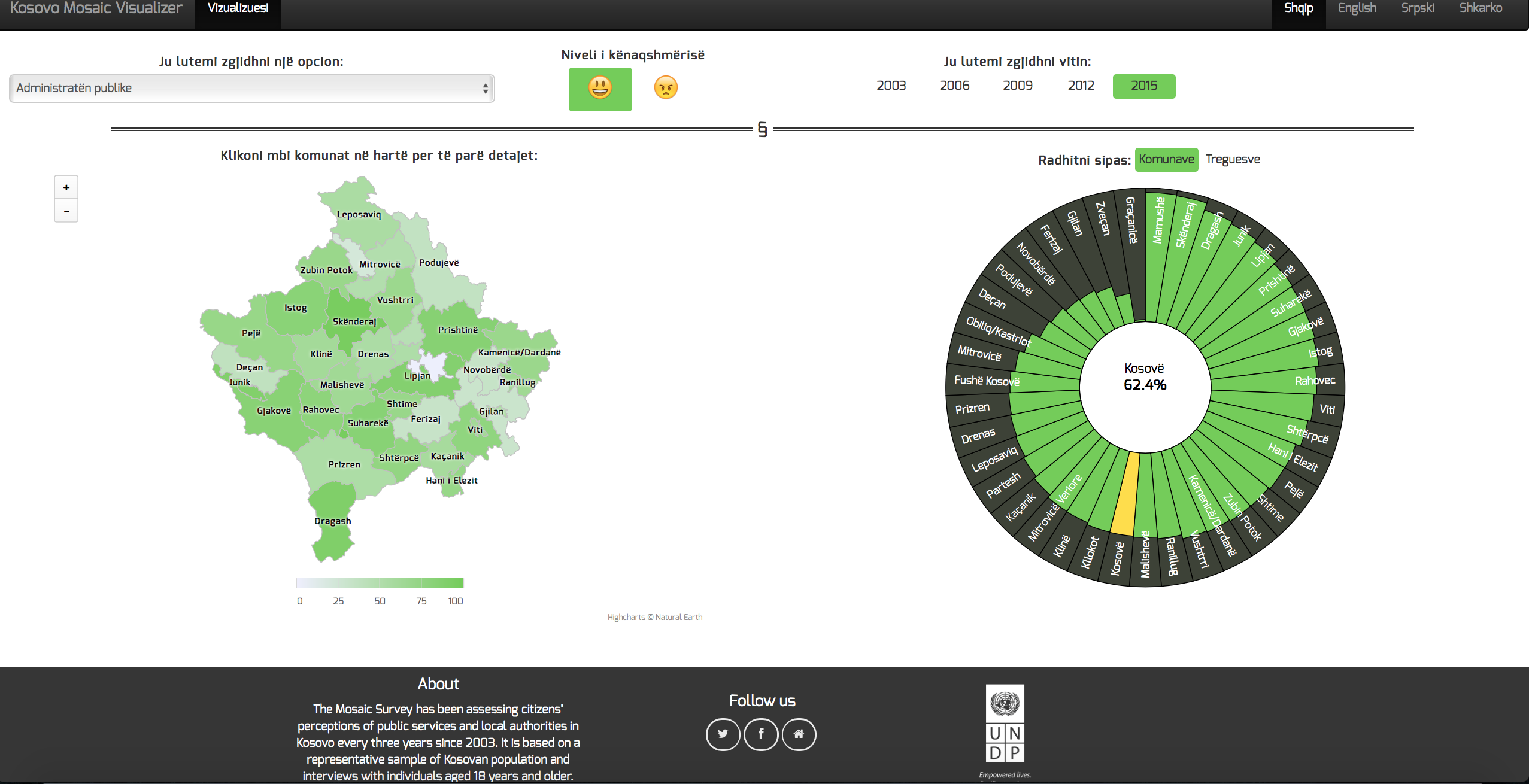
# Visualization Structure

## Variables

Throughout the visualization there are four different dimensions along which the user can interact with the data throughout the visualization. These are:

1. The survey year – 2003, 2006, 2009, 2012 and 2015 (at the time of writing).
2. Satisfied/dissatisfied – The possible responses for respondents were ‘Very Satisfied’, ‘Somewhat Satisfied’, ‘Don’t Know’, ‘Somewhat Dissatisfied’ and ‘Very Dissatisfied’. The ‘Satisfied percentage’ reflects the percentage of people who were ‘Somewhat’ or ‘Very Satisfied’. The ‘Dissatisfied percentage’ reflects the percentage of people who were ‘Somewhat’ or ‘Very Dissatisfied’.
3. The indicator – The indicators are a list of public services and processes about which citizens were asked to provide their level of satisfaction or dissatisfaction.
4. The municipality – The user can view municipality specific data for available municipalities, provided the municipality was involved in that survey.

## Landing Page



The landing page provides the user with options to change the selections for three of the dimensions mentioned above:

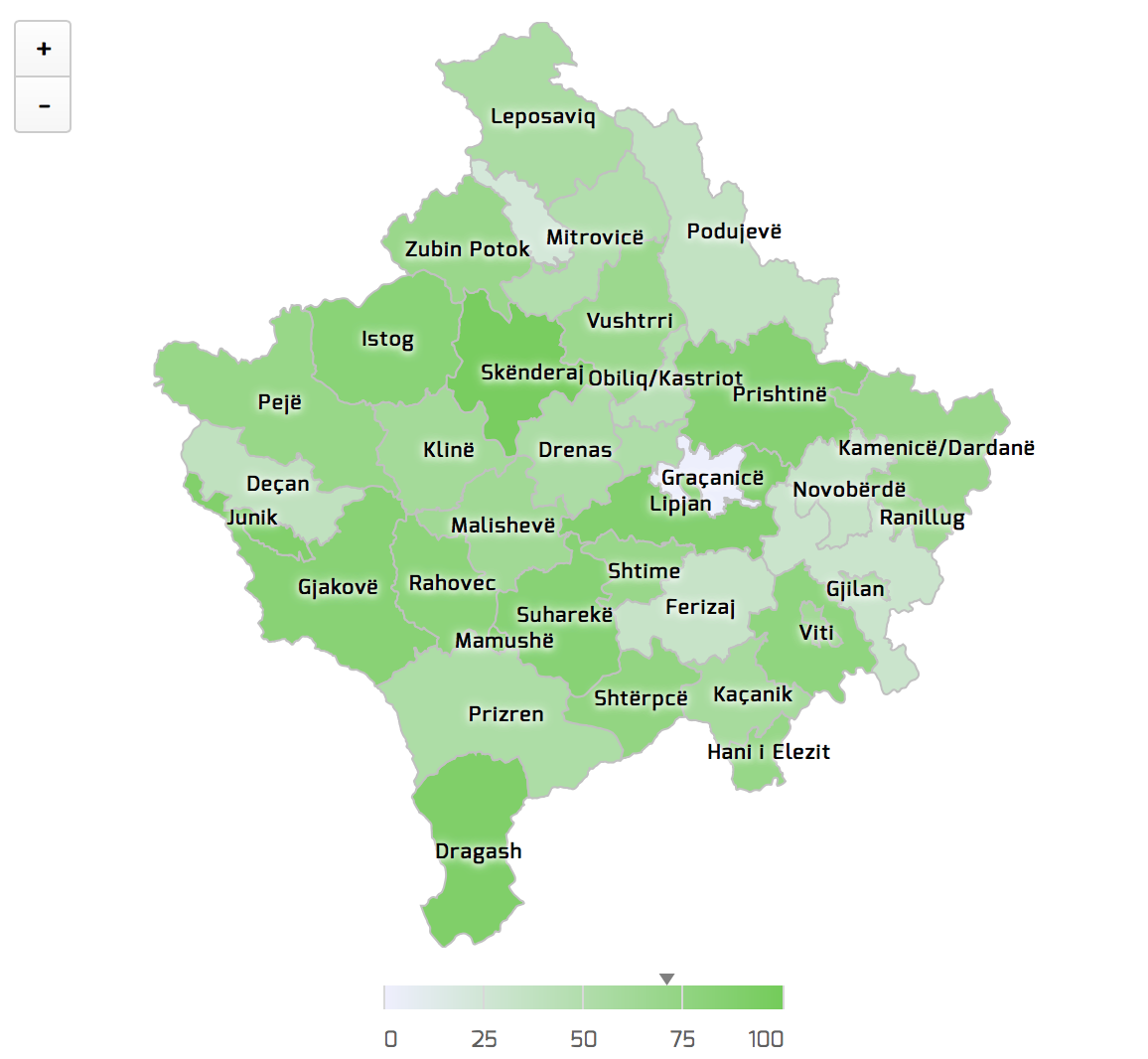
1. The survey year can be selected using the radio buttons in the top right hand corner of the page. Only one year can be selected at a time.
2. To switch between the satisfied (the default) and dissatisfied percentages, there are two face buttons, one happy and one angry, at the top center of the page. Clicking these buttons will switch between the two views.
3. To modify the indicator, the user is provided with a drop down box at the top right hand corner of the page.

Municipality specific data cannot be viewed directly from this page for more than one indicator at a time but, as is detailed below, by clicking a municipality on the map, the user can access more detailed municipality specific information.

On the Landing page this data is displayed using two charts:

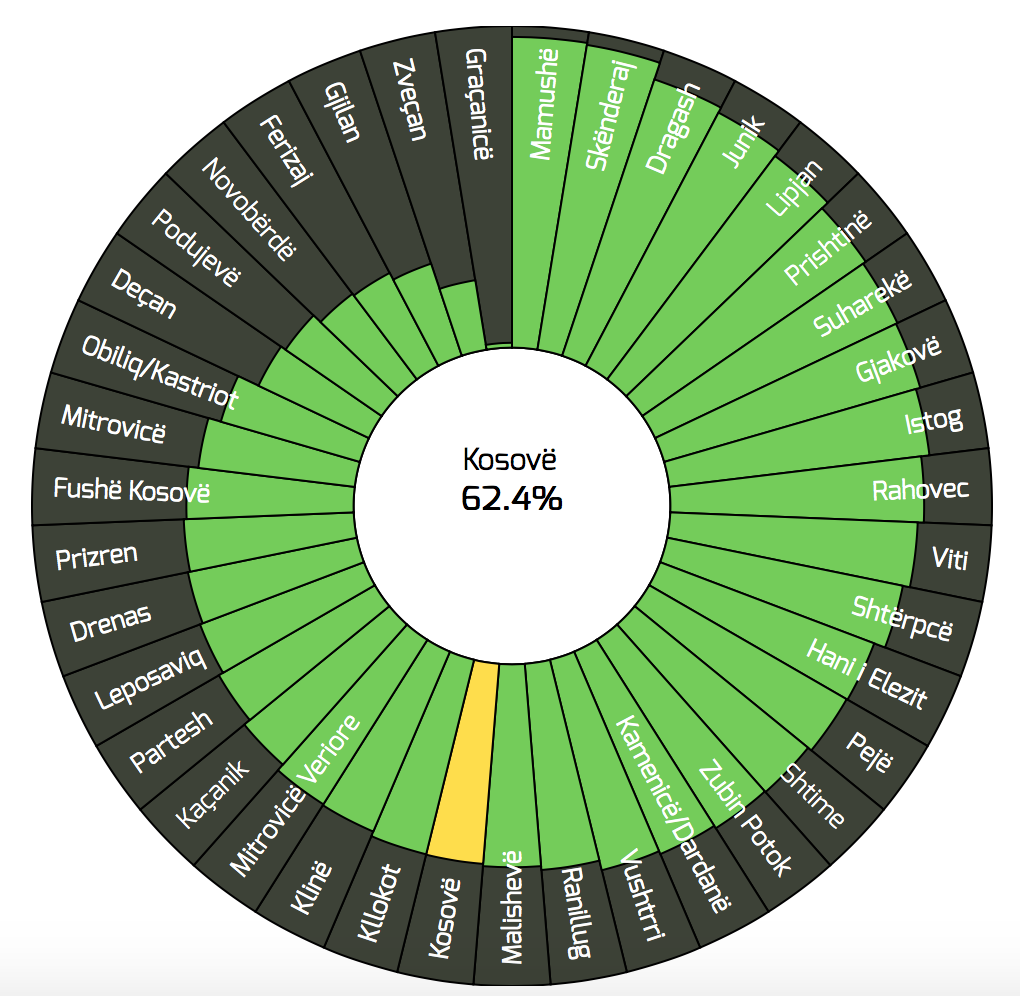
1. An interactive Map of Kosovo (see Figure 1) that displays, for the selected indicator and year, the percentage of satisfied or dissatisfied citizens for each municipality through varying shades of green or red respectively.

#### Figure 1 – Interactive Map of Kosovo

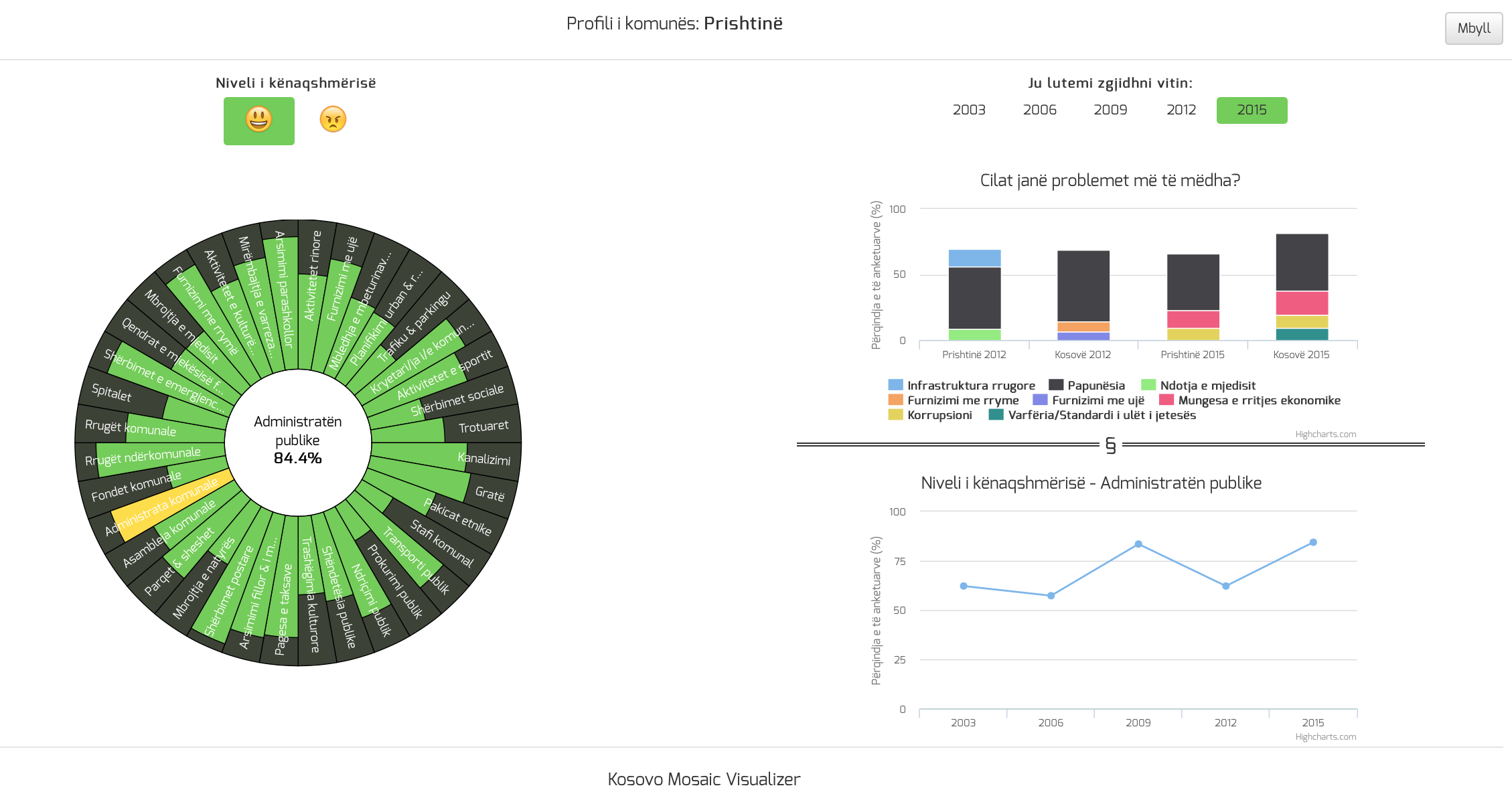


1. An Aster Chart that has two different views:
   1. The default view (see figure 2) displays a ranking of the municipalities for the selected indicator and year, based on the level of satisfaction/dissatisfaction.
   2. The other view displays the Kosovo average level of satisfaction/dissatisfaction across all indicators for the selected year. It should be noted that in this view, selecting an indicator will update the selection as would occur if the user selected it from the dropdown box.

#### Figure 2 – Aster Chart Default Municipality Ranking View



## Municipality Profile Page



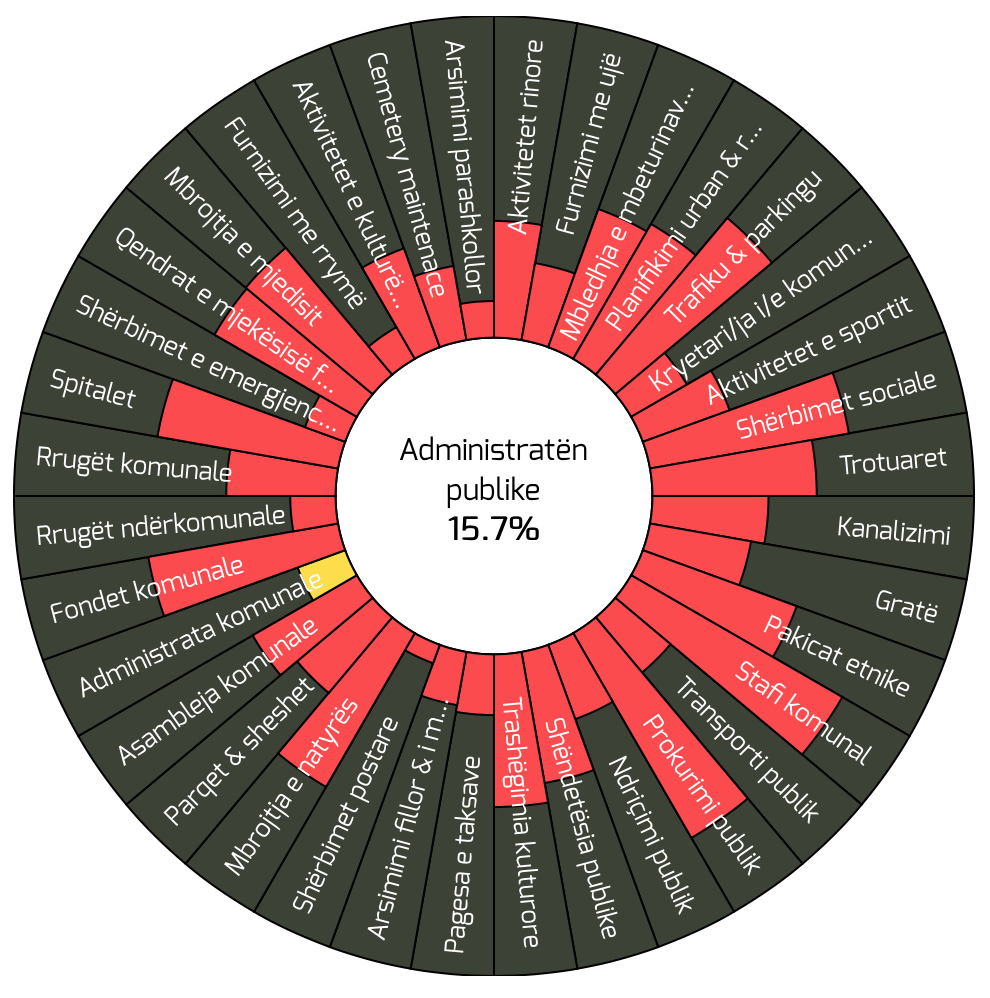
The user can access a second ‘Municipal Profile’ page by clicking on a municipality in the Map. On this page the user has access to similar options as the Landing page for changing their selections. The year selector buttons and satisfaction/dissatisfaction options are similarly located on the Municipality profile page.

One option that is slightly different is that the indicator dropdown is not available on this page. To select a different indicator on the Municipality Profile page, the user selects the indicator on the Aster chart that will then update the Line chart (see below for details).

Overall, the Municipality Profile page displays 3 charts:

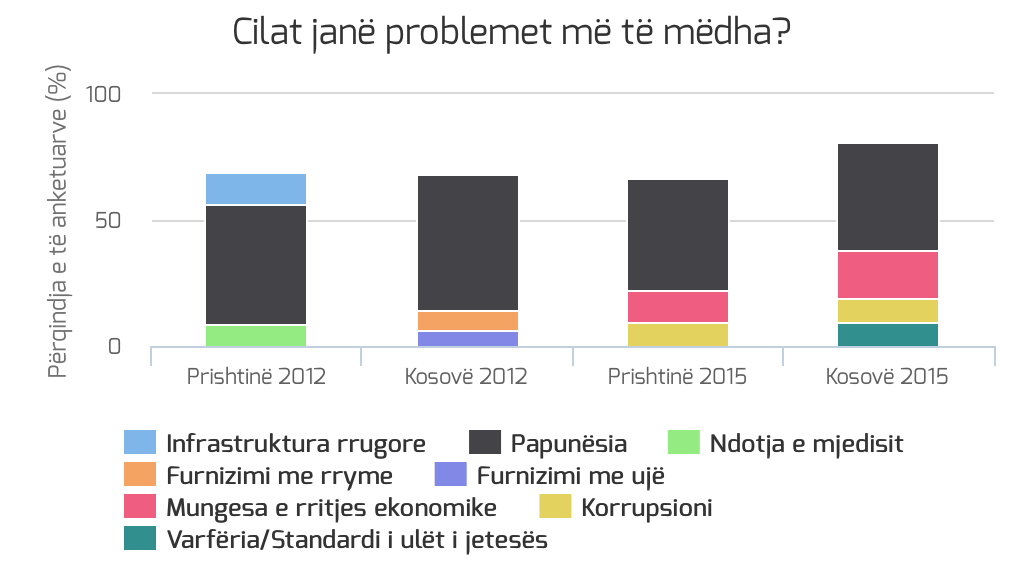
1. A new version of the Aster Chart. This chart now displays the levels of satisfaction/dissatisfaction across all indicators for the selected year for that municipality. As mentioned above, clicking an indicator on this chart will update the indicator selected in the Line chart.

#### Figure 3 – Aster Chart, Municipality Profile page



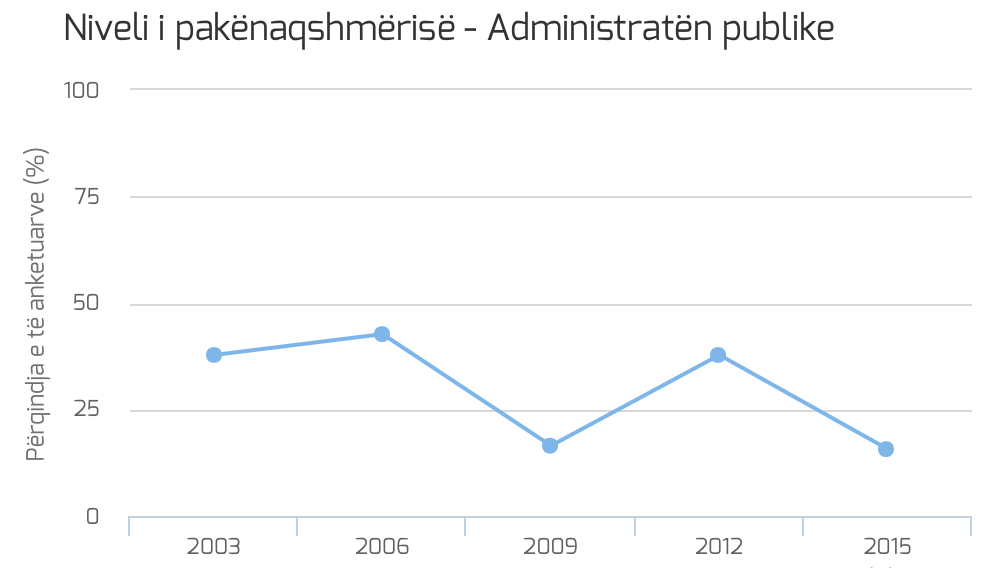
1. A stacked column chart. This chart displays data from the problems dataset and shows the three main things that people perceived to problems in their municipality and the percentage of people who listed them. This is then compared to the Kosovo average for all years available.

#### Figure 4 – Stacked Column Chart, Main Problems



1. A line chart. This chart displays the satisfaction/dissatisfaction level for the selected indicator across all years available. This allows users to see trends in how that indicator has changed over time.

#### Figure 5 – Line Chart, Indicator Satisfied Percent over Time



# Other Features

## Information Pages

### Indicators Page

*To be added.*

### Problems Page

*To be added.*

## Languages

The visualization is available in English, Serbian and Albanian. The language can be changed by using the language buttons on the right hand side of the navigation bar.

## Download the Data

Clicking the download button on the far right of the navigation bar will download all data relevant for the visualization. This includes:

1. Satisfied percentages across all indicators, municipalities and years
2. Dissatisfied percentages across all indicators, municipalities and years
3. Problem data across all municipalities and years
4. The standard lists and translations used for:
   1. Municipalities
   2. Indicators
   3. Problems

# Technology

The visualization runs on HTML and javascript and [Jekyll](http://jekyllrb.com). The Aster chart utilizes the [D3.js](http://d3js.org) javascript library, while all other charts in the visualization use [Highcharts](http://www.highcharts.com).

Repository also includes a Python class and script that is used for processing data. To run this script requires access to the [Pandas](http://pandas.pydata.org) Python library.

# Updating

The Kosovo Mosaic visualization has been designed in such a way that it is as dynamic as possible. In practical terms, this means that by updating the two underlying data files, the visualization should also update and display correctly, provided it is in the correct structure and is processed correctly. The following section provides a guide on how to update the visualization.

## Introduction to GitHub

Currently, the entire visualization is being hosted, free of charge, using GitHub. GitHub is a service that allows users to create public ‘repositories’ or ‘repos’ to store code, free of charge. As of 2015, GitHub added a new service named ‘gh‑pages’. This service provides users with free hosting if the repository is a web site/app and the repository is structured in a certain way.

Assemblio, in creating this visualization, has utilized this service as a way of guaranteeing free hosting and easy maintainability.

### Set Up Git

Depending on your computer setup, you may need to have Git installed on your machine.

##### OSX

Macs generally come with a basic version of Git already installed, but you may wish to upgrade it. The latest stable version can be found here:

<http://git-scm.com/download/mac>

##### Linux

If you are running Linux (and don’t have Git installed already), please use the following instructions to install Git:

<https://git-scm.com/download/linux>

### Cloning the Repository

Once Git is installed, you need to download all the files from GitHub to your computer. To do this, change directory to the folder where you would like the project folder to be located and run the Git command shown below. Keep in mind that Git will automatically create a folder with all the relevant files inside it for you. You don’t need to create an empty folder beforehand.

|  |
| --- |
| git clone https://github.com/assemblio/kosovo-mosaic-visualizer |

### Setting up Access to Update the Repository

Now that you have all the files on your machine, the next step is to set up the access to ‘push’ any changes you make to the online code repository that you just downloaded the files from. To do this, you will need to be granted access by someone who is an administrator of the repository, then run the command shown below.

|  |
| --- |
| git remote add origin https://github.com/assemblio/kosovo-mosaic-visualizer.git |

To verify this has worked, you can run the following command:

|  |
| --- |
| git remote –v  # Expected Output  origin https://github.com/assemblio/kosovo-mosaic-visualizer.git (fetch)  origin https://github.com/assemblio/kosovo-mosaic-visualizer.git (push) |

If you see the expected output shown above, you now have the ability to make changes to the online version of the visualization.

## Step 1. Updating the Data

Generally, the current files should serve as templates for the format of any new updated data. However, there are certain additions that can be made to the files that will update the visualization in the expected format. Broadly, there are two files that need to be updated to update the visualization:

1. data.xlsx
2. whitelist.xlsx

### Step 1a. Update the Satisfied/Dissatisfied Data

The satisfied/dissatisfied datasets are saved in the raw\_data folder and are saved in an Excel file data.xlsx. The full file path is as follows:

* ./data/raw\_data/data.xlsx

The satisfied and dissatisfied data is contained on the tabs labeled ‘satisfied’ and ‘dissatisfied’ accordingly. These tabs can be effectively added to or modified by:

1. Adding new rows to the data. These rows can have new indicators and/or new years, and do not need to have values for all municipalities. If new indicators are added, they also need to be added in whitelist.xlsx.
2. Modifying existing rows. The values, indicators and years can be modified. If the names of existing indicators are being modified, corresponding changes will also need to be made in whitelist.xlsx (see Standard Lists below).

##### Important Notes

1. It is **strongly** recommended that you modify a copy of the original data.xlsx file. This will ensure you can check the format and structure of the original if your updates are not working as expected.
2. The file name (except the ‘.xlsx extension) for the file is flexible – the new names will be handled in the processing step (see Converting).
3. Adding “satisfaction with” or “dissatisfaction with” to the name of the indicator is unnecessary. These are added automatically by the visualization.
4. Format the numbers as decimals rather than as percentages. For example, format the numbers as 0.157, not 15.7%.
5. Do not change the municipality names. Changing the names of the municipalities from the names currently in the file will cause the data for the changed municipalities to not display (the names are required to match names used to generate the Map visualization).
6. Do not use merged cells. Please ensure that you simply repeat the value in the column for all the rows you would typically merge.
7. Do not change the names of the tabs.

### Step 1b. Update the Problems Data

The process for updating/modifying the problems data is very similar to the process for updating the satisfied/dissatisfied data. The problems dataset is saved in the same file as the satisfied and dissatisfied data:

* data/raw\_data/data.xlsx

The tab containing the problems data is labeled ‘problems’. This tab can be effectively added to or modified by:

1. Adding new rows to the data. These rows can have new problems and/or new years, and do not need to have values for all municipalities. If new problems are added, they also need to be added to the standard lists file.
2. Modifying existing rows. The values, indicators and years can be modified. Again, if indicators are being modified, corresponding changes will also need to be made to the standard lists file.

##### Important Notes

1. It is **strongly** recommended that you modify a copy of the original data.xlsx file. This will ensure you can check the format and structure of the original if your updates are not working as expected.
2. The file name (except the ‘.xlsx extension) is flexible – it will be handled in the processing step.
3. Changing the names of the municipalities from the names currently in the file will cause the data for the changed municipalities to not display (the names are required to match names used to generate the Map visualization).
4. Additional problems also need to be added to the standard list files (see below). If new problems are added to the problems data *without* adding them to the standard lists, the visualization will not update correctly.
5. Format the numbers as decimals rather than as percentages. For example, format the numbers as 0.157, not 15.7%.
6. Do not use merged cells. Please ensure that you simply repeat the value in the column for all the rows you would typically merge.
7. Do not change the name of the tab.

### Step 1c. Standard Lists

This step is only required for updating the visualization if new indicators/problems are added to their respective datasets, or the names of existing indicators/problems have been changed. *If no changes have been made to the indicator or problem names, this step is not required.*

The file that needs to be modified in this section is in the following location in the file structure:

* data/standard\_lists/whitelist.xlsx

##### New/Modified Indicators and/or Problems

If new indicators/problems have been added or existing indicators/problems modified, then those changes will need to be reflected in the lists file used by the visualization.

In this file you will find the three standard lists used by the application. The two tabs that you will need to modify are the ‘indicators’ and ‘problems’ tabs.

The data in these tabs reflects the list of all indicators/problems in the visualization and contains the following columns:

1. **Index** – this column contains the *internal* name of the indicator/problem. This name MUST match the name of the indicator used in the satisfied/dissatisfied files.
2. **English** – this is the name of the indicator/problem that will display in the visualization when the user has the English language selected.
3. **english\_short** - (*indicators tab only*) – this is the short form English name for the indicator that will display in the Aster charts
4. **albanian** – this is the name of the indicator/problem that will display in the visualization when the user has the Albanian language selected.
5. **albanian\_short** (*indicators tab only*) – this is the short form Albanian name for the indicator that will display in the Aster charts
6. **serbian** – this is the name of the indicator/problem that will display in the visualization when the user has the Serbian language selected.
7. **serbian\_short** (*indicators tab only*) – this is the short form Serbian name for the indicator that will display in the Aster charts

*If you are modifying an existing indicator/problem, please remember to update the existing row in the list rather than creating a new row.*

##### Important Notes

1. It is **strongly** recommended that you make changes to a copy of the original whitelist.xlsx file. This will ensure you can check the format and structure of the original if your updates are not working as expected.
2. The file name (except the ‘.xlsx’ extension) is flexible – it will be handled in the processing step.
3. Changing the names of the municipalities from the names currently in the file will cause the data for the changed municipalities to not display (the names are required to match names used to generate the Map visualization).
4. Do not change the names of the tabs. They should remain as they are in the original whitelist.xlsx file.

## Step 2. Converting the Files

After all the required files have been updated as described above – the next step is to run a command from the command line that will convert all those files to the formats required for the visualization.

### Step 2a. Installing Requirements

*Please note: This step only needs to be done* ***once*** *for a given computer.*

In order to run the Python script that will convert the files, we need to ensure Python and some key libraries needed for the conversion are available. By creating a virtual environment, we can ensure we have these requirements without having to install them system wide. The command to create this virtual environment (from the project root directory) is shown below. Please be aware that this process can take up to 20 minutes to complete.

|  |
| --- |
| cd bin/  bash install.sh  cd .. |

Once everything has completed successfully (you should see ‘Installation complete’ printed on the final line of output), this command will have installed a virtual environment on your computer that will be used to run a Python script.

### Step 2b. Activating the Virtual Environment

To use the virtual environment, run the following command from the project root directory:

|  |
| --- |
| source bin/venv/bin/activate |

Your command line prompt should now have ‘(venv)’ printed at the front, indicating that you are now operating from this environment.

### Step 2c. Running the Command

The next step is to run the command that will convert the changes made in the Excel files into changes in the files that the visualization uses.

The command that will be run is calling a Python script that will in turn run a purpose built Python class called MosaicData. This command will specify which files to refresh and where the files are that it needs to convert.

This is done through the use of command line parameters. A summary of the available parameters is provided below.

#### Table 2 – create\_data.py Parameters

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| --s | Including this parameter will cause the script to refresh the satisfied indicators data. The parameter should immediately be followed by the file path of the file with the new satisfied indicators data. This file should be an xlsx file with a ‘satisfied’ tab in the same format as data.xlsx. |
| --d | Including this parameter will cause the script to refresh the dissatisfied indicators data. The parameter should immediately be followed by the file path of the file with the new dissatisfied indicators data. This file should be an xlsx file with a ‘dissatisfied’ tab in the same format as data.xlsx. |
| --p | Including this parameter will cause the script to refresh the problems data. The parameter should immediately be followed by the file path of the file with the new problems data. This file should be an xlsx file with a ‘problems tab in the same format as data.xlsx. |
| --w | Including this parameter will cause the script to refresh the standard lists used by the visualization. The parameter should immediately be followed by the file path of the file with the new standard lists. This file should be an xlsx file in the same format as whitelist.xlsx. |

Please note that all the parameters above are optional. The script can be run to update one file only if needed. Below is an example call that updates all the files based on the current default names of the raw files.

#### Sample Call

|  |
| --- |
| python bin/create\_data.py --s data/raw\_data/data.xlsx --d data/raw\_data/data.xlsx --p data/raw\_data/data.xlsx --w data/standard\_lists/whitelist.xlsx |

If the script has run successfully, a success message should be returned confirming which files were updated. This can be further confirmed by running the command ‘git status’. This should provide a list of all the updated files in the directory and should include the original xlsx files modified in step 1, and a list of corresponding js files (the output of the python command above).

### Step 2d. Exiting the Virtual Environment

Once you have finished running the command, you should exit the virtual environment with the following command.

|  |
| --- |
| deactivate |

## Step 3. Testing the Changes

The final test to confirm the changes have worked is to look at the page and see if the data has updated. To do this, we need to have Jekyll installed.

### Step 3a. Installing Jekyll

*Please note: This step only needs to be done* ***once*** *for a given computer.*

##### Mac OSX

To install Jekyll on a Mac, you need to run the following command:

|  |
| --- |
| gem install jekyll |

Depending on your version of OSX, this may return an error like this:

|  |
| --- |
| You don't have write permissions for the /Library/Ruby/Gems/2.0.0 directory. |

If you see this error, add ‘sudo’ before ‘gem’ and run the command again:

|  |
| --- |
| sudo gem install jekyll |

##### Ubuntu

For step-by-step instructions for installing Jekyll on Ubuntu, please see the following:

[Installing Jekyll on Ubuntu](http://michaelchelen.net/81fa/install-jekyll-2-ubuntu-14-04/)

### Step 3b. Starting the Server

The next step is to start the Jekyll server. This should be done while in the root directory of the project folder, with the following command:

|  |
| --- |
| jekyll serve |

This will produce a few lines of text that looks something like this:

|  |
| --- |
| Configuration file: none  Source: /file/path/to/kosovo-mosaic-visualizer  Destination: /file/path/to/kosovo-mosaic-visualizer/\_site  Incremental build: disabled. Enable with --incremental  Generating...  done in 5.468 seconds.  Auto-regeneration: enabled for '/file/path/to/kosovo-mosaic-visualizer'  Configuration file: none  Server address: http://127.0.0.1:4000/  Server running... press ctrl-c to stop. |

The key piece of information needed from this output is the ‘Server address’. This tells us where we can view our modified page before we make the changes permanent.

### Step 3c. Viewing the File

To view the visualizer you need to use the ‘Server address’ as the first part of the URL, add ‘index.html’ and enter the whole thing into your browser. For example – using the address in the example output above, the URL to view the page would be:

<http://127.0.0.1:4000/index.html>

If you have the same server address as in the example, you can just use the link above.

Once you have done this, the page that displays should be the updated version of the visualization based on the updated files. Double check the page works in all languages and that all the functionality is working as expected before moving onto Step 4.

Once you have viewed your changes, press control + c to stop the server in the terminal window.

### Undoing All Changes – The Nuclear Option

Sometimes it happens that your updates will not work and you cannot remember how to undo all the changes you made. In this case the best option maybe to undo *all* your local changes and simply go back to the version already online and start again. To do that, run the following commands while in the root directory:

|  |
| --- |
| git fetch origin  git reset --hard origin/master |

*Keep in mind, this will delete all your changes and you will not be able to recover them.*

## Step 4. Pushing the Changes Online

Now that you have updated the changes on your computer, and the testing has shown everything is working as expected, you need to push those changes to the online version. The following are the steps to do that:

### Step 4a. Push to master branch

|  |
| --- |
| git add -A  git commit –m “Enter a description of your changes here”  git push origin master |

### Step 4b. Push to gh-pages branch

|  |
| --- |
| git checkout gh-pages  git merge master  git push origin gh-pages |