University of Zurich

Master Basic Module

Implementation of Heat Maps in the Swiss Feed Database with Google API

Sajan Srikugan
Msc. UZH in Informatics, Software Systems

supervised by

Prof. Dr. Michael Böhlen
Dept. of Informatics – Database Technology

June – August 2019

Table of Contents

1. Introduction

For visualization and analysis purposes, Agroscope uses the Swiss Feed Database to store, update and visualize new information about nutrient measurements of different feeds. The main feature on the web application are heat maps, visualizing the density of samples and the regression from nutrient variables. There is also the possibility to visualize the heat map by setting additional filters like location, elevation, feeds, time, season or nutrients. Other key features are statistical tools like a scatter plot, visualizing the distribution of nutrients over several years and a data table.

In this report, the current problems of the new version of feedbase.ch will be explained and elaborated. Then the requirements from Agroscope relative to heat map visualization and stability of the current version will be summarized and conducted. After having given some information about the implementation details, it will be continued with the analysis of heat map data visualization via Google Maps API.

1.1 Current issues in the Swiss Feed Base

Since the old version of feedbase.ch is not maintained anymore and does not properly visualize the data, a new working version for Agroscope is needed. The new version based on Google API was already online, but it had the following difficulties:

- cantons have no color assigned
- rendering of the map causes a crash of the browser
- by design: static heat map in all zoom levels
- interpretation of many results were not possible, since the influence on a single kernel bandwidth of a single data point was too high. That lead to a fully red heat map in some cases
- no location markers of any sample was displayed
- inconsistent map reset when switching between nutrients or regression and density map
- no heat map visualization in certain nutrient measurements of feeds
- by design: no map displayed when top query has no location data available

The bug-fix of these basic functionalities will be elaborated in chapter 3. The most of these bugs were explored during the project work and needed to be fixed first. Some bugs were also mentioned in the first requirements session with Agroscope. Originally, the main focus had to be on the visualization of the heat maps. But since some main functionalities of the new feedbase.ch version didn’t work, it had to be fixed first before approaching to the heat map visualization.

2. Requirement Analysis

At the beginning of the first session with Agroscope, some general issues of the current version were discussed. The main problem is the heat map visualization, which was not even nearly the same when comparing some feeds in the old with the new version. On some regression examples it was showed, that nearly all visualizations were not interpretable. In top queries with many data available, the whole midland in Switzerland was colored red (high values) in the new feedbase version, while the data was partially blue (low values) in the old feedbase.ch version. This makes it rather impossible for Agroscope to work with the current heat map. This was also illustrated with screenshots and comparisons from their side.

On a local client installation, some basic functionalities were explained. During this session, first not working functionalities were explored and discussed. This was also the case in subsequent sessions, where new malfunctions in the software have been discovered during the session. The fixing of these malfunctions was then also a part of the requirements.

2.1 Overview of what was fixed and what implemented

For a better overview, the fixing of several software bugs and the implementation of requirement ideas are illustrated in the following tables. Details about the fixing and implementation can be found in chapter 3.

Description	**Fixed**	**Not fixed**
No color assignment in cantons	x	
Rendering of map causes a crash of the browser	x	
No location markers available	x	
No data retrieval when switching between regression entities	x	
Overlay of heat maps when filtering with side menu	x	
Map reset after radius search		x
By design: asynchronous loading of canton, sample density and regression layer		x

Figure 1: Software bugs

A radius search can be made by the user by zooming in three times or more in the cantonView layer. By clicking a location marker, there is the possibility to draw a radius from the clicked location marker and only visualize the data within the circular area. But if the user switches to the regression or density layer after drawing the circle and tries to clear the radius, a map reset is not properly done. That means the heat map of the circular area is still present.

Another issue is the asynchronous loading of the canton, density and regression layer. This is due to the design of the code. If the user clicks on a top query and toggles the regression category before the default cantonView is loaded, an overlapping of the two categories will be concluded. An idea to avoid this behavior is a loading window, which locks the map until the corresponding data is completely loaded.

Description	**implemented**	**Not implemented**
Dynamic heat maps in all zoom levels	x	
Reduction influence radius of data points	x	
Continuous display of cantonView in all zoom levels	x	
Info message when no location data is available	x	
After the cantonView layer, more detailed district layers should be displayed		x
No heat map transitions at water zones		x
No heat map transitions at certain altitudes		x

Figure 2: Implementation of requirements

Requesting and retrieving data can take some time, especially if there are many data points available for a certain top query. Since it is still a browser application, an introduction of more detailed district layers can cause some difficulties with the rendering of the map and therefore a crash of the browser. Due to performance reasons, it is rather not recommended to introduce district information. Another reason is the fact that there are many cantons like Geneva, Zug, Uri etc. who don’t know the division into districts. In that sense an implementation of district layers has no great additional value for the end user, since district borders are only available for a few cantons.

By design, a top query always starts with a cantonView layer on Google Map with zoom-level 0. Therefore, a rough overview of the data is provided. If there shouldn’t be any heat map transitions on water zones or at certain altitudes, there would be an unnatural cut of the heat map and therefore holes at lakes surrounded by many measurement points (e.g. lake of Zurich, lake of Zug, lake Neuchatel) in the overview map on zoom level 0. This is aesthetically not pleasing for the user and is maybe not following the principal of a heat map. For the altitude idea there is already a filter in the side menu, where additional altitudes could be admitted.

3. Implementation

3.1 Assignment of color for cantons

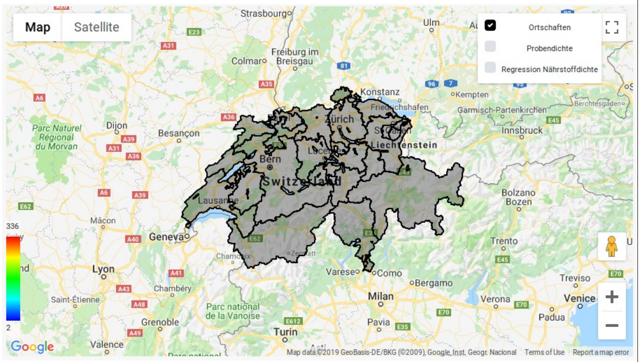
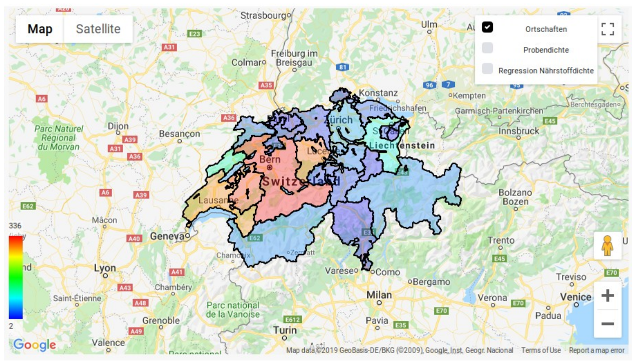
	
Canton layer with wrong “fillcolor” assignment	Canton layer with correct “fillcolor” assignment

Figure 3: color assignment for cantons

The Data Style Option of Google Maps API has the following key properties: fillColor, fillOpacity, strokeColor, strokeOpacity, strokeWeight and title. To assign colors to each canton, one has to call the getproperty() method, which returns the value of the requested property, or undefined if the property does not exist. Since the fillcolor property only supports HTML color codes in hexadecimal form, the assigned RGB values needs to be transformed by the getShapeColor() function to hexadecimal color values. The assigned RGB value of a canton is dependent on the number of recorded data (density). The RGB assignment works as follows:

R(ed):	**G(reen):**	**B(lue):**
if density <= 0.5 return 0;	if density >= 0.25 return 255;	if density <= 0.25 return 255;
if density >= 0.75 return 255;	if density < 0.25 return (density/0.25*255)	if density >= 0.5 return 0;
else (density-0.5)/0.25*255)	else (1-((density-0.75)/0.25)) *255);	else (1-((density-0.25)/0.25)) *255);

Figure 4: RGB assignment for cantons

That means that cantons with a higher absolute number of probes tends to result in a red and the opposite case tends to result in a blue coloring. All values in between received a light blue, green, orange or yellow color schema. In this case, an intuitive color spectrum was chosen.
If a canton has no data available, then no color and no stroke for the particular canton is going to be assigned. This means that the canton shape is not marked in any way (e.g. Geneva in the example above).

```
$scope.getCantons = function() {
   $http.post('/api/map/cantons', FilterParams)
     .then(function(res) {
       try {
         $scope.map.data.addGeoJson(res.data);
         $scope.map.data.setStyle(function(feature) {
           return {
             fillColor: $scope.getShapeColor(feature.getProperty('quantity_normalized')),
             title: feature.f1,
             strokeWeight: 2
           }
         });
```

Listing 1: Fixing of color assignment

3.2 Displaying location markers of samples

The location markers are defined by longitude and latitude. These two variables are used to determine the exact location of the marker on Google Map. The display of the location markers was already optimized by only retrieving the distinct locations. But on a low zoom level, the map does not provide a good overview, since the canton layer is hidden by many markers.

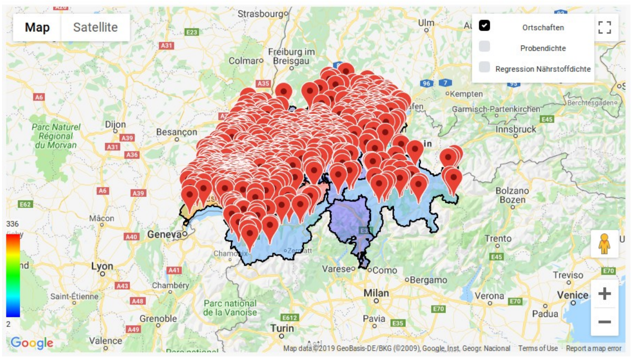
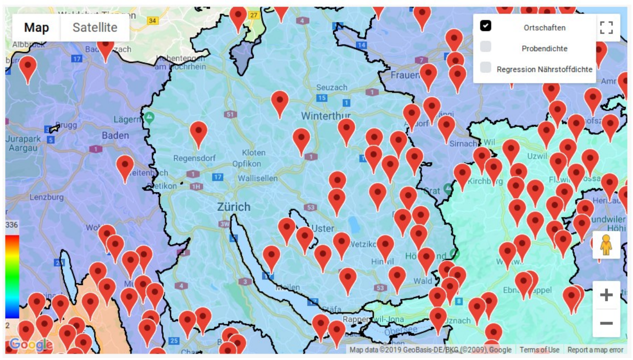
 Map Satellite Ortschaften Probendichte Regression Nährstoffdichte 336 2 Google	 Map Satellite Ortschaften Probendichte Regression Nährstoffdichte 336 2 Google
Location markers on zoom-level 0	Location markers on zoom-level 3

Figure 5: location markers on different zoom levels

For that reason, the markers are only displayed from zoom-level 3 and higher. This leads to a clearer map. The location markers are not being displayed at zoom level 0,1 and 2. As soon as the user goes back to zoom-level 2, only the cantonView Map will be loaded and the location marker cleared. In accordance to Agroscope, the cantonView layer should always be displayed, independent of the current zoom level.

```
$scope.computeOverlays = function() {
  FilterParams.params.mapZoom = $scope.map.getZoom();

  if ($scope.mapOverlayOptions.showMarkers) {

    if (FilterParams.params.mapZoom > 9) {
      $scope.cantonViewActive = false;
      $scope.getLocations();
    } else if (FilterParams.params.mapZoom <= 9 && !$scope.cantonViewActive) {
      $scope.cantonViewActive = true;

      // Reset Map Bounds
      $scope.resetMapBounds();

      $scope.clearLocations();
      $scope.getCantons();

    }
  }
};
```

Listing 2: location markers only from a certain zoom level on

3.3 Dynamic Heat Maps and kernel density of data points

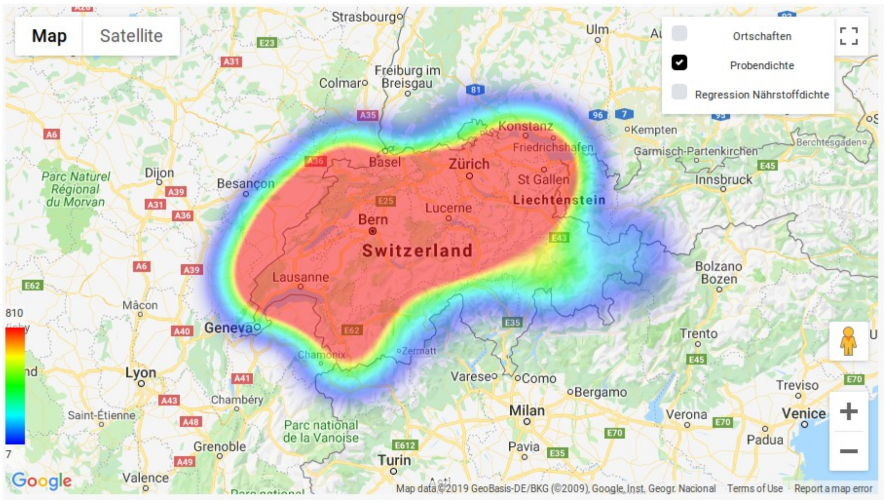


Figure 6: Regression heat map on a low zoom level

In case many data were available, an interpretation of the results was not possible. Since the heat map was also static, a higher zoom level results in the same issue. This is due the fact that the influence on a single kernel bandwidth of a single data point was too high. That lead to a fully red map in some cases. In agreement with Agroscope, it was decided to change the heat map to a dynamic behavior. A dynamic heat map was also not the case in the old feedbase.ch version, where a static kernel regression of the data was used. A dynamic solution gives more flexibility for analyzing the data in every zoom level.

In the sample density heat map implementation, for each data point is given a radius of influence of 8000 meters. The choose of the radius has a big influence on the visualization of the heat map. Here some comparisons are shown for the example “Heuernte 2017”:

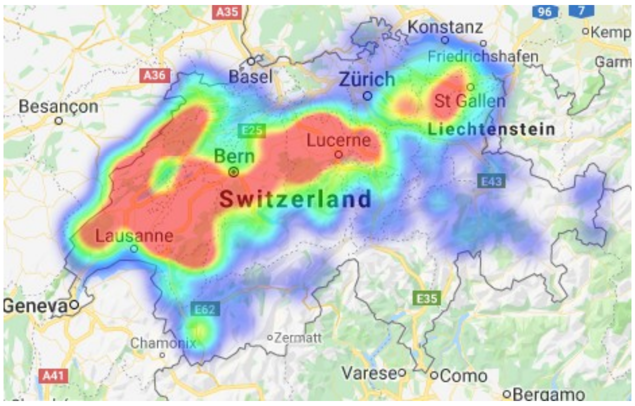
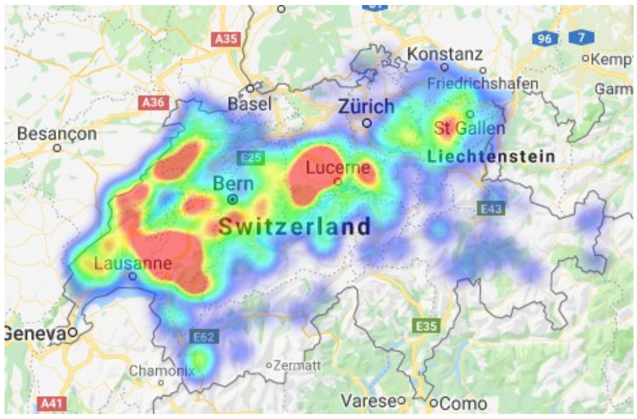
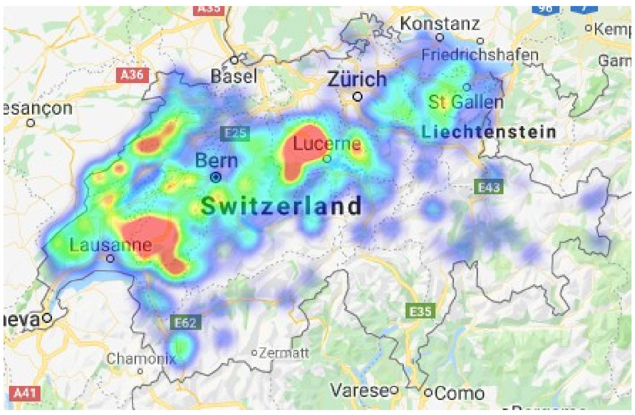
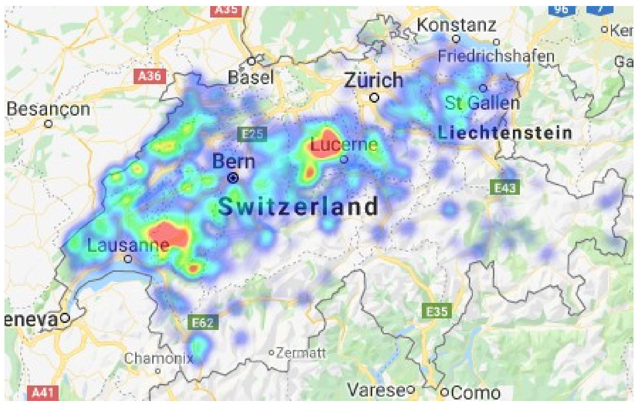
	
$scope.HEATMAP_RADIUS_METERS = 12000;	$scope.HEATMAP_RADIUS_METERS = 10000;
	
$scope.HEATMAP_RADIUS_METERS = 8000;	$scope.HEATMAP_RADIUS_METERS = 6000;

Figure 7: Influence of setting a radius in sample density map

Since the amount of data is variable for each query, there is no best-case practice for the choose of the radius. By default, heat map colors are dynamically scaled according to the greatest concentration of points at any particular pixel on the map. Unfortunately, this causes some issues, which will be elaborated in chapter 4. There is also a property “maxIntensity” that allows to specify a fixed maximum. This can be helpful if the data consists a few outliers with a high density. Also the case of categories with only a few data needs to be considered. If the radius is chosen to small, then there is a smaller scope of interpretation when lowering the influence of a single data point. This is the case if a query is filtered by a canton or the measurements was only taken on few specific locations.

For the visualization of the regression heat map, a different radius for the data points were used. Since the band width of the data between max- and min-value is much smaller compared to the sample density heat map, a small difference on the absolute value of the calculated data can result in a different color assignment. Here an example on the content on Phosphor for “Heuernte”.

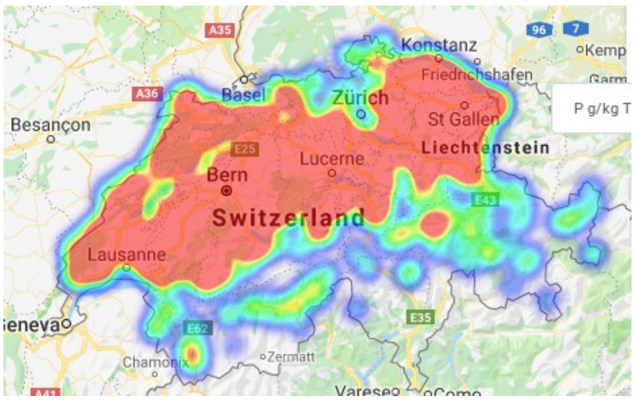
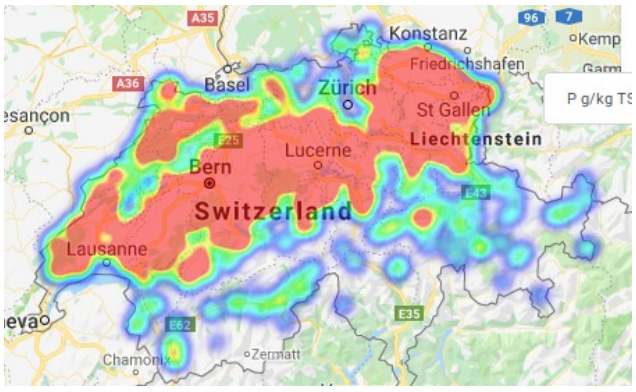
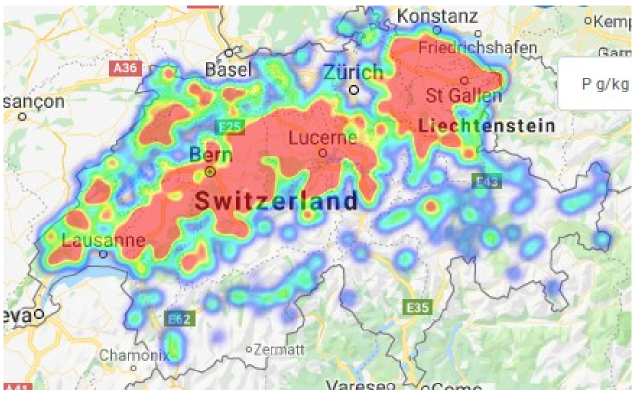
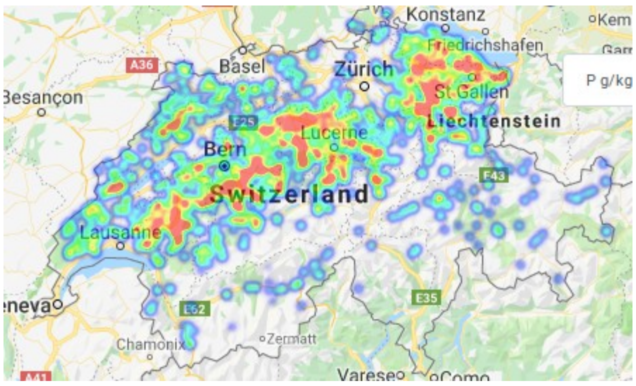
	
$scope.HEATMAP_RADIUS_METERS = 10000;	$scope.HEATMAP_RADIUS_METERS = 8000;
	
$scope.HEATMAP_RADIUS_METERS = 6000;	$scope.HEATMAP_RADIUS_METERS = 4000;

Figure 8: influence of setting a radius on visualization in density map

Currently, the radius was fixed at 5500. Since there were pictures from the old version, a similar visualization for Phosphor was given with a radius of 5500.

3.4 No location data available message

In case the filtered data table does not have a location and PLZ value, there is no longitude/latitude returned to the client. This leads to an empty, blank window with no map. To avoid confusion, it was decided to display a information in this case. For the sake of completeness, the Google Map should be always displayed and not only if there is location data available. Therefore, a transparent black layer was placed above the map with a white info message:

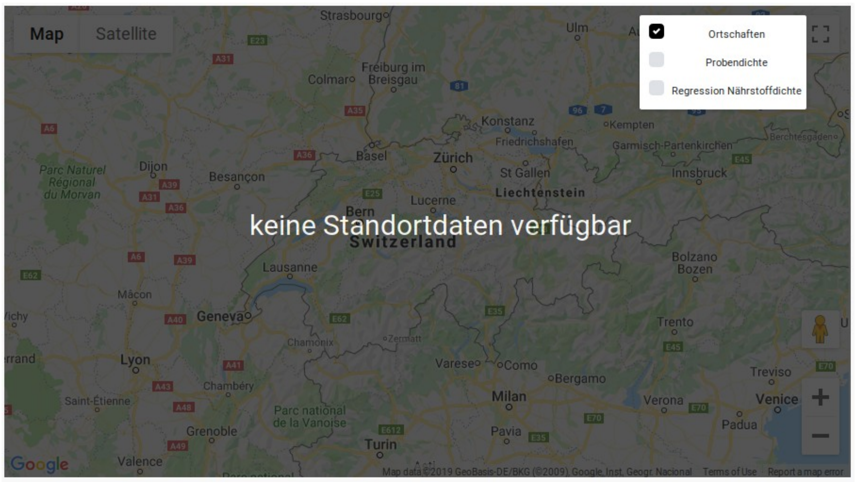


Figure 9: information message in case of no location data

// main.css code for the transparent black layer	// main.css code for the information display

```
.location-data-error-message {
  height: 100%;
  display: flex;
  justify-content: center;
  align-content: center;
  z-index: 1;
  background-color: #111;
  width: 100%;
  position: absolute;
  opacity: 0.8;
 }
```
	```
	.location-data-error-message div {
	  font-size: 35px;
	  position: absolute;
	  top: 220px;
	  height: 50px;
	  width: 100%;
	  color: #fff;
	  left: 270px;
	 }
	```

Listing 3: css code for info message

Visualization code only if there is no map/location data

```
div(id='visualization-wrapper' ng-class=`{'filter-menu-offset': sideBarVisible}` ng-cloak)
  div(class='row display-flex')

    div(id='map-wrapper' ng-controller='MapController' class='col-lg-6')
      div(class="location-data-error-message" ng-show="showMapErrorMessage")
        div No location data available
```

Listing 4: visualization code in backend

3.5 Switching between nutrients does not display data

One of the main issues in this project was the missing regression data when switching between regression entities. For every entity, a SQL-query is executed on the server site to retrieve the corresponding data. By enabling the tracing of executed SQL-queries in pgadmin, it is possible to check, which statements are performed. In the table below, there is an example for the content of calcium in swiss hay:

```
WITH regressionNutrient AS (
   SELECT tmp.latitude      AS latitude,
          tmp.longitude     AS longitude,
          Count(*)::integer         AS count,
          Avg(tmp.quantity) AS quantity,
          MIN(lims_number) AS lims_number
        FROM   (SELECT Avg(quantity) AS quantity,
          lims_number,
          latitude,
          longitude
         FROM   d_feed,
          fact_table_clean,
          d_nutrient,
          d_origin,
          d_time
         WHERE  fact_table_clean.id_feed_fkey = d_feed.feed_key
          AND fact_table_clean.id_nutrient_fkey = d_nutrient.nutrient_key
          AND fact_table_clean.id_origin_fkey = d_origin.origin_key
          AND fact_table_clean.id_time_fkey = d_time.time_key
          AND latitude IS NOT NULL
          AND longitude IS NOT NULL
          AND canton IS NOT NULL
          AND (id_feed_fkey IN (3,12,2,1)) AND (id_nutrient_fkey IN (88)) AND id_nutrient_analyses_fkey IN
('2') AND ((d_time.moment = 1 OR d_time.moment = 2) AND d_time.t_year IN ('2017')) AND
d_nutrient.nutrient_key IN (88) AND fact_table_clean.id_nutrient_analyses_fkey IN (11)
         GROUP  BY lims_number,
             latitude,
             longitude) AS tmp
        GROUP  BY tmp.latitude,
           tmp.longitude
     )
   SELECT * FROM regressionNutrient
```

Listing 5: wrong SQL-query for retrieval of calcium regression data

By executing this statement directly in pgadmin, no results were returned. This is also the reason why no heat map was displayed on the client site, since there is no returned value. By checking the where-clause, it is directly clear that only tuples from the fact_table_clean table with the attribute value 2 AND 11 in “id_nutrient_analyses_fkey” can be returned. Since there is no other table with the attribute id_nutrient_analyses_fkey, both parts of the where-clause are referring to the same table. But the fact that the value for “attribute_id_nutrient_analyses_fkey” must be 2 AND 11 is clearly a contradiction, since there is only one integer value for each tuple allowed and therefore mutually exclusive.

By checking the other regression entities for the same top query “Heuernte”, it was visible that “id_nutrient_analyses_fkey IN ('2')” was always there in every query of all entities. The data of a particular entity was only retrieved, if the other AND-clause fact_table_clean. id_nutrient_analyses_fkey IN () was also “IN 2”, thus identical. There was a workaround for this problem, which was also used to identify a solution for the problem. If we filter out the content of calcium in swiss hay in the side menu, only the data table for the entity calcium is being displayed and therefore only heat map data for this entity is visualized. In this case, the data retrieval works correctly:

```
WITH regressionNutrient AS (
   SELECT tmp.latitude      AS latitude,
         tmp.longitude     AS longitude,
         Count(*)::integer         AS count,
         Avg(tmp.quantity) AS quantity,
         MIN(lims_number) AS lims_number
      FROM   (SELECT Avg(quantity) AS quantity,
         lims_number,
         latitude,
         longitude
        FROM   d_feed,
         fact_table_clean,
         d_nutrient,
         d_origin,
         d_time
        WHERE  fact_table_clean.id_feed_fkey = d_feed.feed_key
         AND fact_table_clean.id_nutrient_fkey = d_nutrient.nutrient_key
         AND fact_table_clean.id_origin_fkey = d_origin.origin_key
         AND fact_table_clean.id_time_fkey = d_time.time_key
         AND latitude IS NOT NULL
         AND longitude IS NOT NULL
         AND canton IS NOT NULL
         AND (id_feed_fkey IN (3,12,2,1)) AND (id_nutrient_fkey IN (88)) AND id_nutrient_analyses_fkey IN
('11') AND ((d_time.moment = 1 OR d_time.moment = 2) AND d_time.t_year IN ('2017')) AND
d_nutrient.nutrient_key IN (88) AND fact_table_clean.id_nutrient_analyses_fkey IN (11)
        GROUP  BY lims_number,
           latitude,
           longitude) AS tmp
      GROUP  BY tmp.latitude,
          tmp.longitude
    )
   SELECT * FROM regressionNutrient
```

Listing 6: correct SQL-query for retrieval of calcium regression data

By checking the getWhere-function in the editor, the following part caused the problem:

```
// if (this.selectedAnalyses) {
// conditions.push(`id_nutrient_analyses_fkey IN (${this.selectedAnalyses})`); }
```

Listing 7: commented out part of regression SQL-query

In case of the top query “Heuernte 2017”, the nutrient ADF (acid detergent lignin) was selected and therefore should have the id_nutrient_analyses_fkey = 2. But it is not clear what the intention of the above if-statement is. Commenting out the mentioned part does not influence the heat map of the already working entities.

3.6 Map reset when changing nutrients or filtering

When switching between nutrients in the side menu or between regression entities, the map from the previous filter selection is not reset. This results in overlapping of several heat maps, when doing a new filter selection.

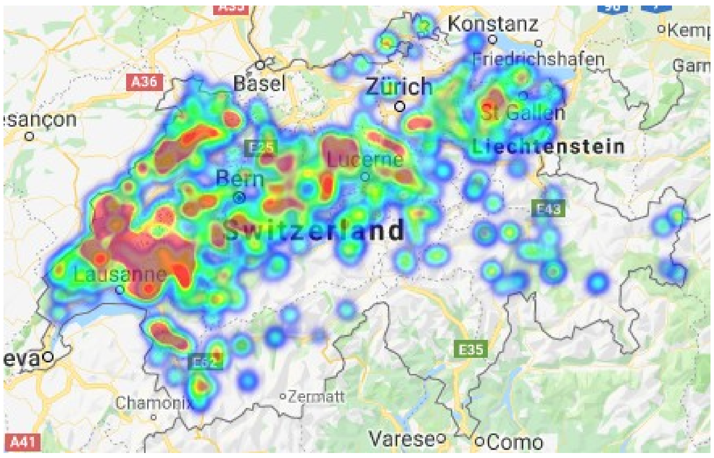


Figure 10: Overlapping of regression heat maps when map reset is inconsistent

Before fixing this issue, there was also an important drawback from Google Maps API mentioned by Valentin Weiss (2018, 16): “The heat map scales each point’s radius according the current zoom level i.e., when zooming out a points radius is increased. This is due to Google Maps handling each point’s radius relative to the map viewport by default. This may be desirable in some scenarios but, in this use case, leads to an unclear picture when zooming out from the map.” This problem can also be replicated when the user zooms to a higher zoom level in the regression map and switches to the density map (or the other way around). In this case, Google Maps API is taking the zoom-level from the first map as current zoom level and displays the density map with a higher kernel radius from the beginning:

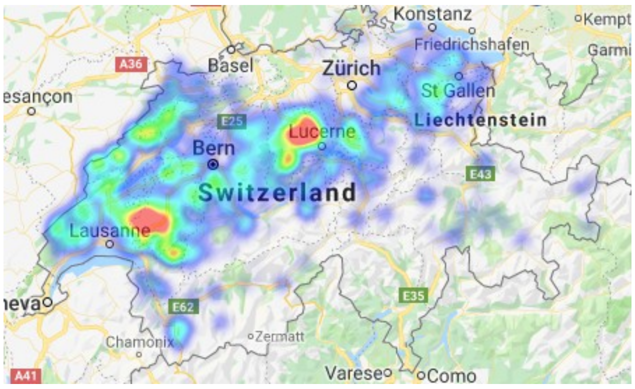
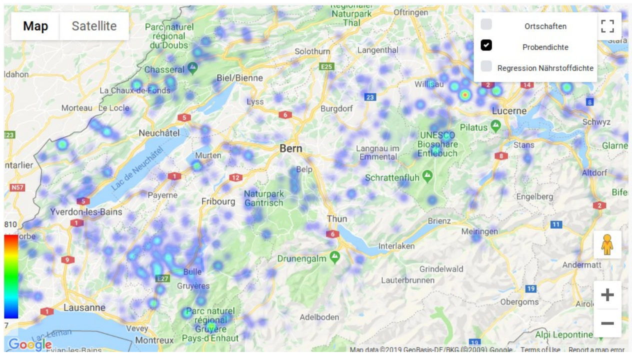
Google Map zoom level: 0 Heat Map zoom level: 0	Google Map zoom level: 2 Heat Map zoom level: 2
	

Figure 11: density map of “Heuernte”

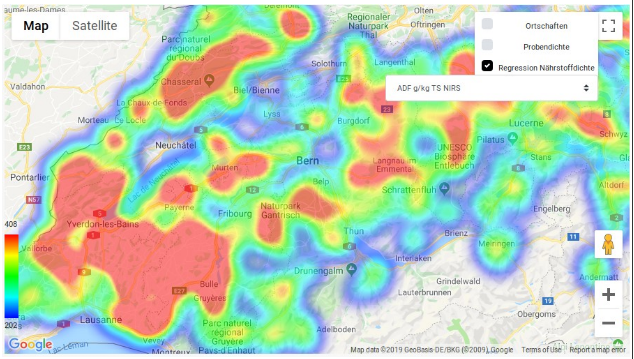
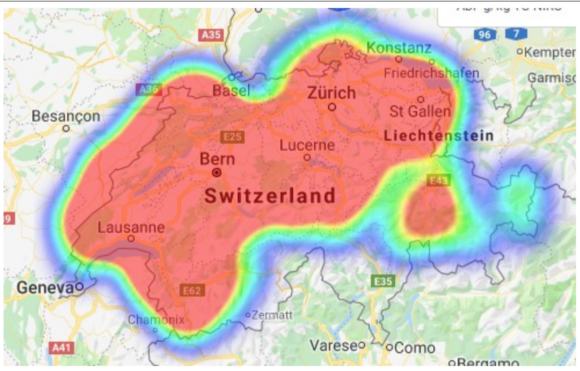
Google Map zoom level: 2 Heat Map zoom level: 0	Google Map zoom level: 0 Heat Map zoom level: -2
	

Figure 12: switch to regression map of “Heuernte” for entity ADF

This drawback should be considered, when doing a map reset. To resolve both mentioned issues, a complete map reset was coded and also an automatically zoom-level reset when switching between regression entities or between regression and density map. Unfortunately, this leads to an “overkill” of GET-Requests for the display of Google Map and thus to a bit poorer performance in the browser. But the poorer performance is only noticeable when a low amount of data needs to be loaded to get a visualization. In case of many data for a top query is available, the GET-request is still faster than the retrieval and calculation of the heat map.

Regression Variables

```
$scope.HEATMAP_RADIUS_METERS = 5500;
// InitMap when changing between Regression variables and reload of Google Map
$scope.map = new google.maps.Map(document.getElementById('map'), {
zoom: FilterParams.params.mapZoom,
center: {lat: 46.797895, lng: 8.2314794}
});

$scope.addColorLegend(stats.min, stats.max);
$rootScope.map = $scope.map;
```

Listing 8: Regression variables

Heatmap Variables

```
$scope.drawSampleDensityHeatMap = function() {
$scope.clearLocations();
$scope.map = new google.maps.Map(document.getElementById('map'), {
zoom: FilterParams.params.mapZoom,
center: {lat: 46.797895, lng: 8.2314794}
});

$scope.HEATMAP_RADIUS_METERS = 7000;
```

Listing 9: Heat map variables

4. Analysis

4.1 Heat Map Visualization on a higher zoom level

For the analysis part, it is important to know that only data with location data are visualized in the heat map. Due to optimization reasons, only distinct locations are visualized. In case of calcium in Swiss hay, there are 4260 recorded data thereof 554 without any location data. If there are several measurements for a distinct location, then the mean of all the data of the particular location will be calculated and returned in the map. This means out of 3706 data points, 1106 of them are visualized as a heat map, since some locations have up to 60 measurements in the Swiss hay category.

In the following table, the returned heat map value for the location "Le Chenit L'Orient" (PLZ: 1341, altitude: 1013 meters) will be illustrated:

Date	**PLZ**	**Kanton**	**Futtermittel**	**Ca g/kg TS**
2014-10-3	1341	Vaud	Heu 1. Schnitt (2)	9
2014-10-3	1341	Vaud	Emd, 2.ff Schnitt (12)	17.8
2014-7-17	1341	Vaud	Heu 1. Schnitt (2)	5.5
2014-9-24	1341	Vaud	Emd, 2.ff Schnitt (12)	10.5
2015-7-7	1341	Vaud	Heu 1. Schnitt (2)	6.3
2015-9-9	1341	Vaud	Heu / Emd gemischt (1)	17.6
2012-1-1	1341	Vaud	Heu 1. Schnitt (2)	5.1
2012-1-1	1341	Vaud	Emd, 2.ff Schnitt (12)	6
2013-9-20	1341	Vaud	Heu 1. Schnitt (2)	17.82
2013-9-20	1341	Vaud	Emd, 2.ff Schnitt (12)	9.53

Figure 13: data table for le chenit l'orient

Returned values of "Le Chenit L'Orient" in the console for the regression heat map:

Count	lims_number	Latitude	Longitude	Quantity	Quantity normalized
10	107-2014-50102161	46.60115000	6.23921000	10.515	null

Figure 14: Returned values for le chenit l'orient in the regression heat map

Even though three samples in le chenit l'orient have one of the highest value in the whole data table (17.8, 17.6, 17.82) concerning the amount of calcium, "only" a 10.515 is returned in the heat map. Since this value is a bit higher than the mean of the complete data (7.546), but far away from the max value of the complete data (20.6), the data point results in a green one:

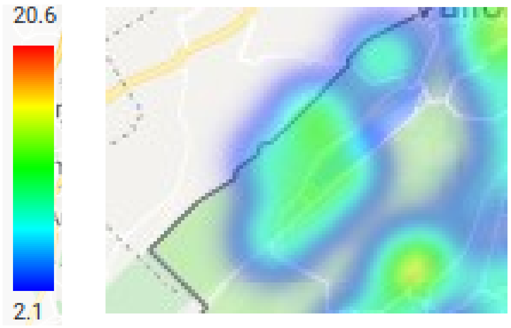


Figure 15: scale and color assignment for le chenit l'orient

Since a single location can have up to 60 measurements, calculating and visualizing the mean of all measurements in this particular location can result in some surprising results at first glance. But this examples shows, that the color visualization on a higher zoom level seems to be correct, based on the current calculation of the returned value.

4.2 Heat Map Visualization on a low zoom level

A bigger issue mentioned by Agroscope is an incorrect visualization from their point of view. In case of the content of calcium in swiss hay, there must be correlation with the altitude of the village. The higher the altitude of a location, the higher the calcium content in Swiss hay. This cannot be confirmed by viewing the new feedbase version. But it can be confirmed in the old feedbase version. Here a visual comparison between the old and the new feedbase.ch version:

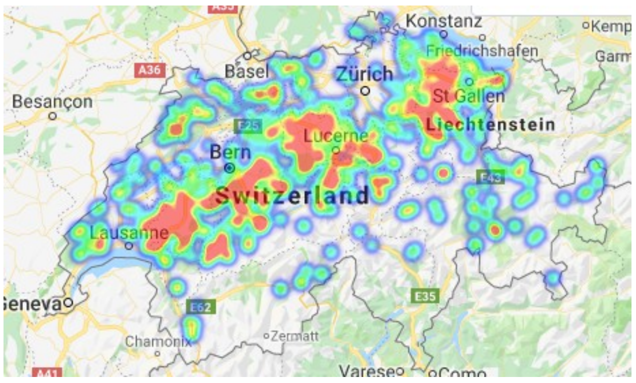
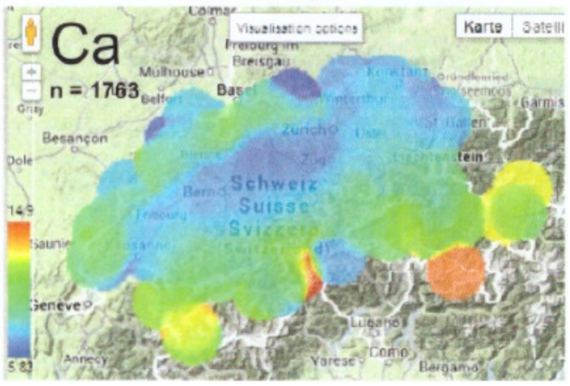
Calcium content of swiss hay between 2010 - 2013 in new feedbase.ch version	Calcium content of swiss hay 2013 in old feedbase.ch version
 Konstanz Friedrichshafen Basel Zürich St Gallen Liechtenstein Besançon Lucerne Bern Switzerland Lausanne Geneva Chamonix Zermatt Varese Como	 Ca n = 1763 Karte Satelli Visualisation options 149 5.83 Besançon Basel Zürich Schweiz Suisse Svizzera Genève Lugano Como

Figure 16: comparision calcium content in swiss hay in new and old feedbase version

Here some basic statistical key numbers:

	New feedbase.ch	Old feedbase.ch
Max value	15.749	14.9
Min value	2.1	5.83
mean	6.960	unknown
Number of data	1838	1763

Figure 17: statistical keys for content of calcium in old and new version

To have an nearly exact comparison, the Swiss hay data from 2010 until 2013 was chosen to have a similar amount of data available. Also the minimal and maximal value are similar, so no big differences in terms of the heat map visualization should be expected. But unfortunately, that's exactly not the case.

According to the new feedbase visualization above, the eastern part of Switzerland and the complete midland should have the highest content of calcium in Swiss hay. By doing a radius search analysis for the mentioned regions above, it will be investigated if the visualization on a low zoom level is correct or not.

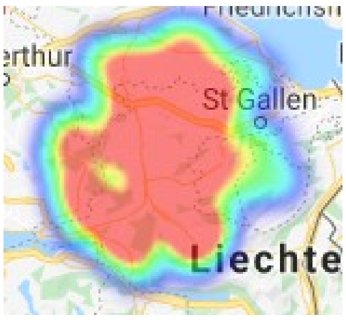
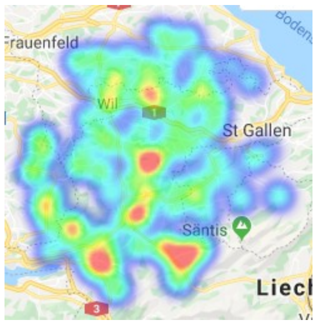
Eastern Switzerland			
Number of data	215		
mean	6.935		
min-value	4.5		
max-value	9.25		
		Zoom-level 0	Zoom level 1

Figure 18: regression analyses for eastern Switzerland

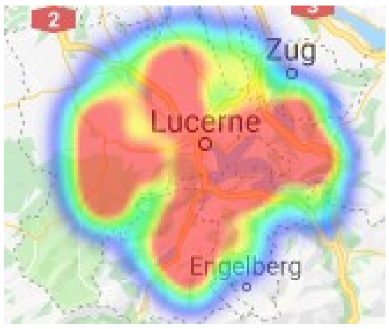
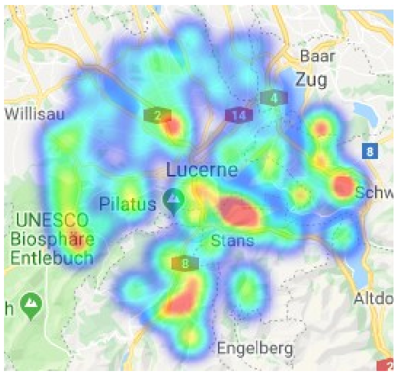
Midland (Lucerne)			 Baar Zug Willisau Lucerne Pilatus UNESCO Biosphäre Entlebuch Stans Schw Altdo Engelberg
Number of data	297		
mean	6.407		
min-value	4.751		
max-value	10.75		
		Zoom-level 0	Zoom-level 1

Figure 19: regression analyses for midland (Lucerne)

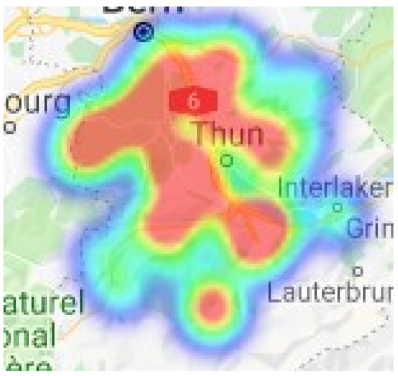
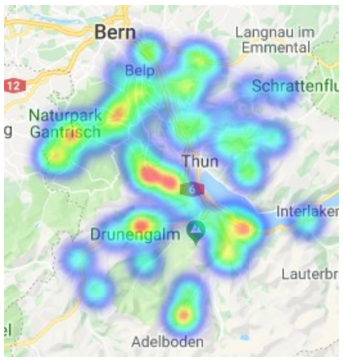
Foothils of the Alps (Thun)			
Number of data	131		
mean	7.062		
min-value	4.6		
max-value	10.883		
		Zoom-level 0	Zoom-level 1

Figure 20: regression analyses for foothils of the alps (Thun)

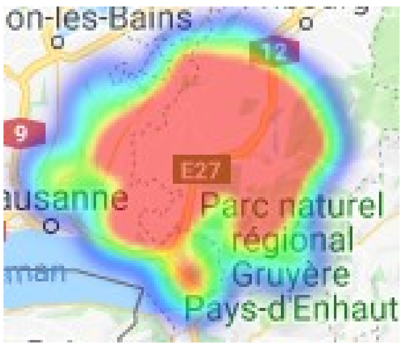
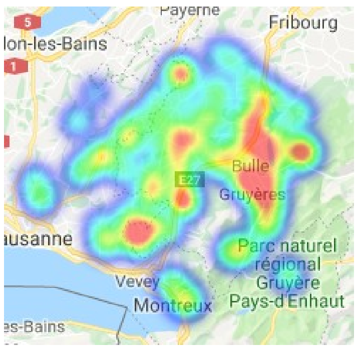
Romandie			
Number of data	281		
mean	6.622		
min-value	2.1		
max-value	12.2		
		Zoom-level 0	Zoom-level 1

Figure 21: regression analyses for romandie

If we consider again the full Swiss hay map with a max-value of 15.749, then there is clearly a visualization issue. Since the mean of the taken examples is around 6.5 – 7, the assigned overall color must be green. This fact arises the question, if Google Maps API visualizes the data based on a density map instead of a weighted point map. Note that the visualization of the regional examples is correct on a higher zoom-level (1 or higher).

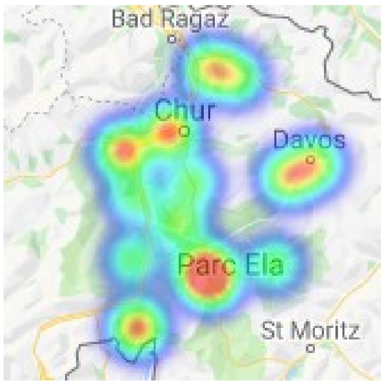
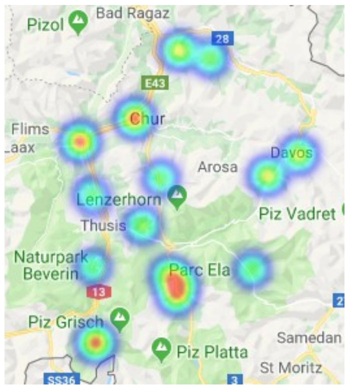
Alps (Obervaz)			
Number of data	21		
mean	10.085		
min-value	6.714		
max-value	12.639		
		Zoom-level 0	Zoom-level 1

Figure 22: regression analyses for alps (Obervaz)

If we consider “Obervaz” in the complete Swiss Map, it’s only visualized as a greenish surface, although the mean (10.085) is much higher than in other parts of Switzerland. This confirms the impression that Google Maps API takes the intensity of available data in a certain region to do the heat map visualization instead of the data values.

4.3 Heat Map Visualization with Google Maps API

The official documentation from Google Maps API describes heat maps as follows:

“A heatmap is a visualization used to depict the intensity of data at geographical points. When the Heatmap Layer is enabled, a colored overlay will appear on top of the map. By default, areas of higher intensity will be colored red, and areas of lower intensity will appear green.”

This description is exactly the case when checking the visualization of the new feedbase.ch version. Regions with many data available (eastern Switzerland, surrounding area of Lucerne etc.) are more intense and therefore colored red. Regions with less data available (e.g. Obervaz) are less intense and therefore colored green. This means that Google Maps API is not applying a “typical” heat map framework on a lower scale. The heat map is using a density approach, which is dependent on the number of data in a specific region. By checking some posts about this issue on stackoverflow, one realizes it’s a known problem. As solution, an implementation over another API was proposed or giving the data weighted values. But Google Maps API is still density based on a lower scale level, although the returned data values in feedbase.ch are weighted points. This can also be confirmed by reading through discussions on stackoverflow. The sources can be found in the bibliography at the end of this report.

4.4 Possible Solutions?

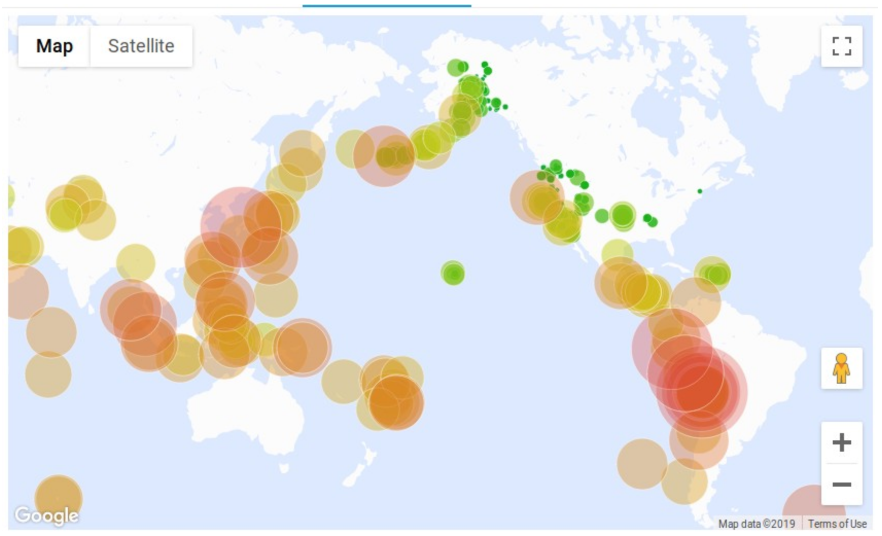


Figure 23: Google Maps API, Advances Style Data Layer

A possible and similar solution to the old feedbase.ch version is an advanced style data layer. The data is visualized as circles with varying radius. Dependent how large the data value is, it also gets a corresponding coloring. In the example above, the magnitudes of all earth quakes in the pacific fire ring is visualized. An earthquake with a high magnitude results in a red circle with a larger radius. The other case are lower magnitudes, which is colored green with a small radius. This could be also implemented in the new feedbase.ch version. If we take the calcium example from chapter 4.2 as instance, there would be many green circles with a small radius in the middle land but also a few large red cirlces, which would stand out in the overview map. The more circles are drawn in a certain location, the more intense is the overlapping color.

Another solution would be to drastically lower the influence radius of a single data point in the current Google Map heat map framework and make the overall map static again. In chapter 4, it was figured out that the visualization on a lower scale is correctly implemented. But this leads to very small spots in locations where nearly no data is available, especially in the mountain regions. This makes it rather hard to interpret the data on a low zoom-level.

5. Conclusion and remaining issues

At the beginning, several software bugs in the new feedbase version and implementation ideas from Agroscope were explained and elaborated. At the end of the project, the main functionalities were fixed and some ideas implemented. Details about the implementation were discussed in chapter 3. The implementation was made in such a way that it satisfies the requirements from a user point of view. Also some drawbacks of Google Maps API were considered and tried to code around. This was for example the case when fixing the map reset by including the viewport drawback. Two issues from the software side still persists. By doing a radius search and switching to the regression or density layer after drawing the circle, a map reset is not properly made when clearing the radius. The heat map of the circular area is still present in this case. The other one is the asynchronous loading of the canton, density and regression layer. If the user clicks on a top query and toggles the regression category before the default cantonView is loaded, an overlapping of the two categories will be concluded.

The biggest issue is still the visualization of the heat maps based on Google Map API. Since the data points on a low zoom level are visualized based on the intensity of available data and not on the values, it is not applying a typical heat map framework. This leads to high differences when comparing certain entities in the new feedbase version with the old one. In that sense two possible solutions were proposed: One would be visualizing the data based on an advanced style data layer and the other would be drastically lowering the influence of a data point and make the overall map static again. Nevertheless, it is possible to work with the heat maps on a higher zoom level. Also the main functionalities are fixed and can be used to work with the web application.

6. Bibliography

[1] Google Maps Platform. (2019). Heatmap Layer. Retrieved August 26, 2019, from https://developers.google.com/maps/documentation/javascript/heatmaplayer

[2] Google Maps Platform. (2019). Data Layer: Earthquakes. Retrieved August 26, 2019, from https://developers.google.com/maps/documentation/javascript/examples/layer-data-quakes

[3] Valentin Weiss. (2018). The Swiss Feed Database, Development of a Dynamic Web Application

[4] Stackoverflow. (2014). Google Heat Map intensity based on value. Retrieved August 26, 2019, from https://stackoverflow.com/questions/24865158/google-heat-map-intensity-based-on-value

[5] Stackoverflow. (2013). Heatmap based on average weights and not on the number of data points Retrieved August 26, 2019, from https://stackoverflow.com/questions/14461954/heatmap-based-on-average-weights-and-not-on-the-number-of-data-points/18873801#18873801

[6] Stackoverflow. (2012). Heat map on Google maps based on the intensity of a parameter. Retrieved August 26, 2019, from https://stackoverflow.com/questions/10742074/heat-map-on-google-maps-based-on-the-intensity-of-a-parameter