Report: CT-Bike

Project for VU Visualisierung at University of Vienna

Bernhard Schatzl, Fuk Pin Chen, Patrick Mittendorfer, Robin Titz

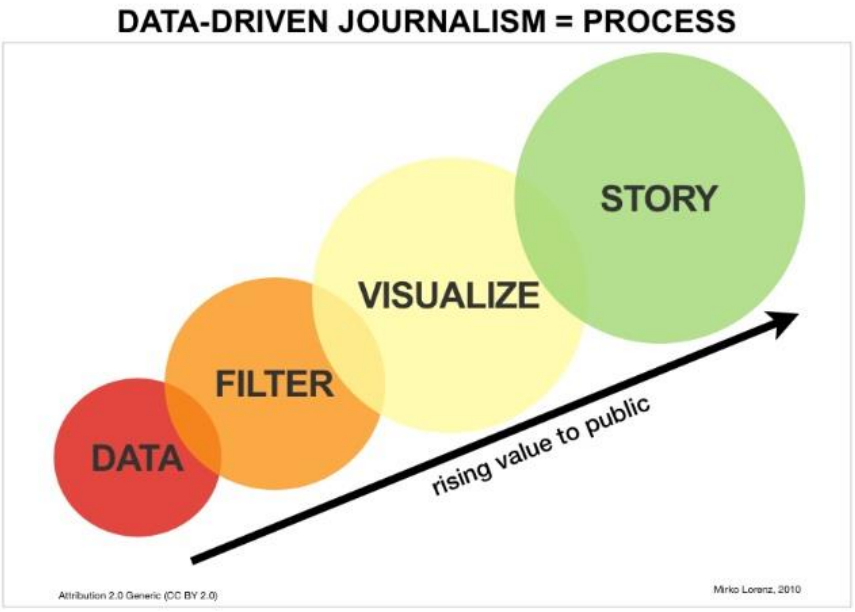
30.6.2015

Motivation

For our visualization project, we've chosen the "Analysing Citybike Data" topic, because all of us are using the city bike service. In addition we really like the concept of sharing services like cars, bikes and other shareable resources.

In this project, we wanted to build a Web Application of the city bike service which helps users to explore the data interactively and intuitively. Our goal is, that the application can be used for data-driven journalism. We are provided with a complex data set of the city bike service. Our challenge is now to observe the data and visualize important questions.

When we've been developing and evaluating our possible target users, we decided, that we won't limit our visualization to a certain user group, but rather provide a visualization for all the people out there, which are curious and willing to consume information about the city bike service. Summarized we could say, that our visualization is based on the ideas of a data driven journalism report.



Our Task is to visualize the Data in a way, which is easily accessible and understandable for the user group. First of all, we have to make the data-sets understandable for us by filtering the data (Using a C#-Tool). After we have the provided data sets, it is our task to make it look attractive for the users. Therefore the user can explore our tool and make his own storyline.

Related Work

We searched on the internet for several existing attempts to visualize city bike data in different cities around the world. Our visualization was not influenced by this research, because most of the other visualizations were provided with a dataset which completely differs from ours.

CitiBike in NewYork

New York also offers a CityBike service called Citibike, for which they provide a real time API. This API (https://citibikenyc.com/stations/json/) provides all the data of the stations and all of the bikes. So you know in real time which bike was just taken and which bike was just brought back.

There are a lot of visualizations of this dataset in the internet, but most of them are showing the driven routes in a time-lapse.

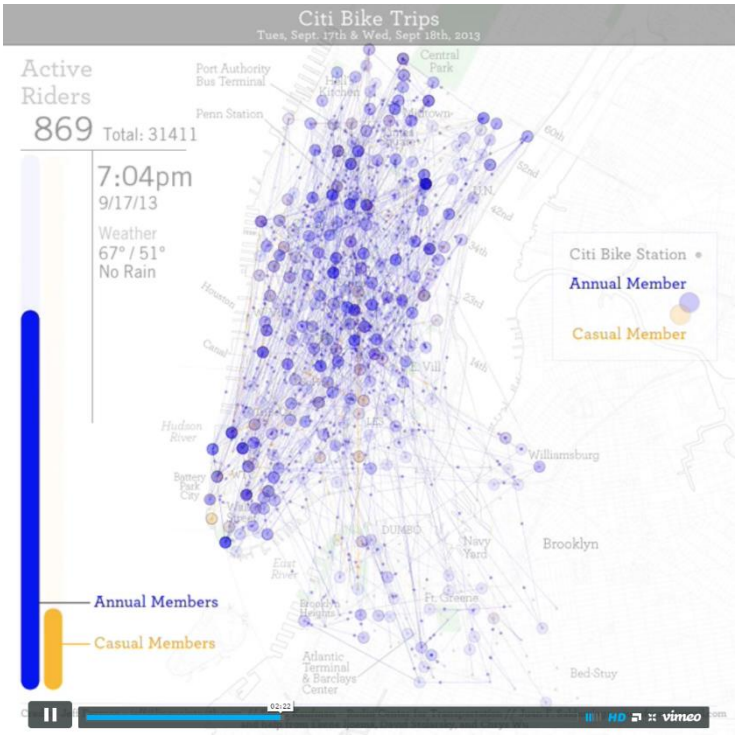
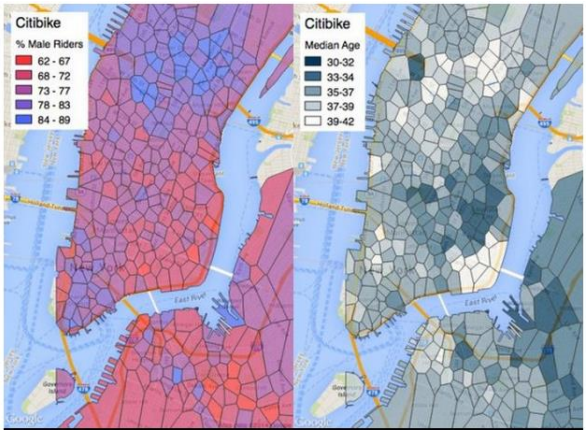


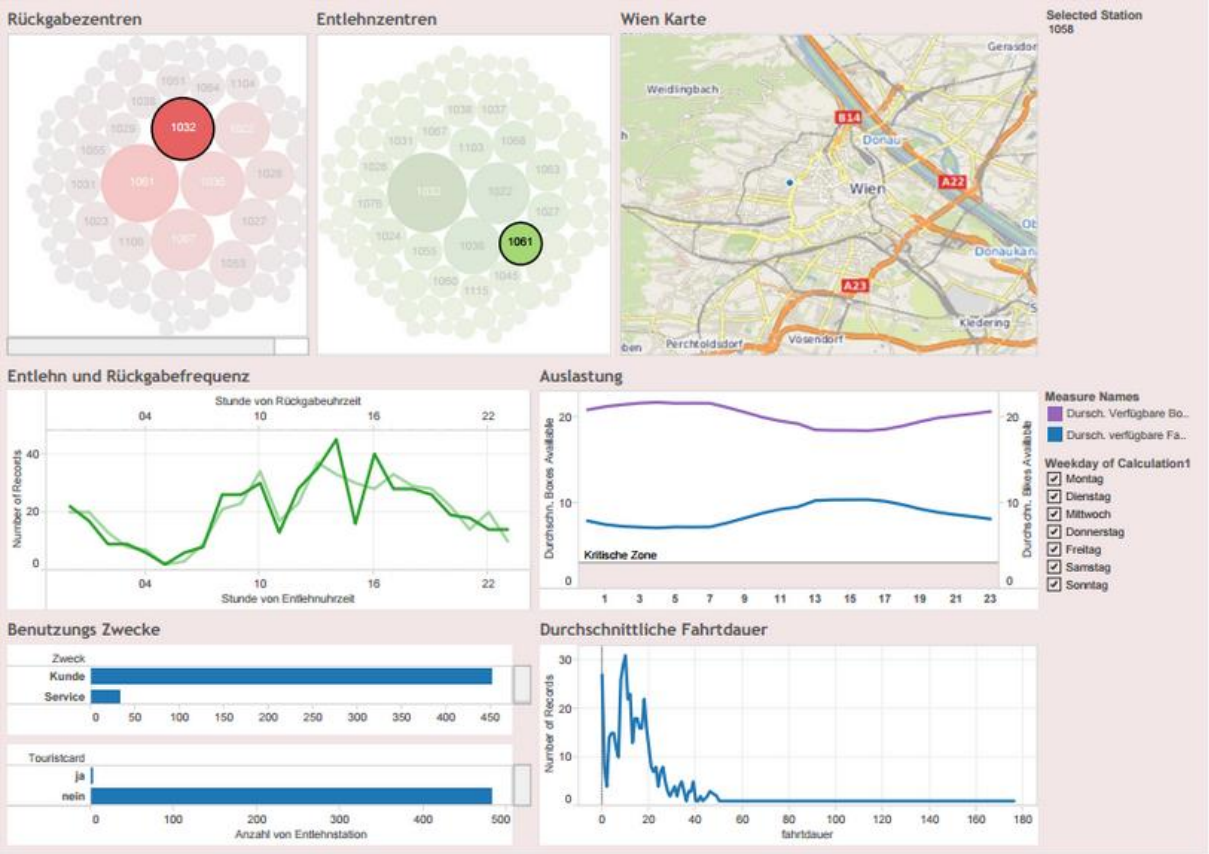
Figure 1: http://linepointpath.com/111242/2771111/work/citi-bike-visualization

In this figure, we can see a map with all the city bike stations. This video shows us how the city bike users are using the service. How many city bike users have rent a bike right now, where are they driving, how long does it take. As mentioned before there are a lot of visualizations like this.

Another visualization analysed the city bike data depending on the user information. They are showing the ratio between male and female costumers and the age of those.



Related work from our course, “citybike visualisation”



They have another solution attempt. First of all they have made their app with a tool called tableau, which helps for making nice looking charts. In the app there are more views. The user can filter via weekdays. In our app it is filtered by months. The best way for the two apps would be to filter both ways, weekdays as well as month. In addition this visualization is provided with different charts, which can’t be found in our tool.

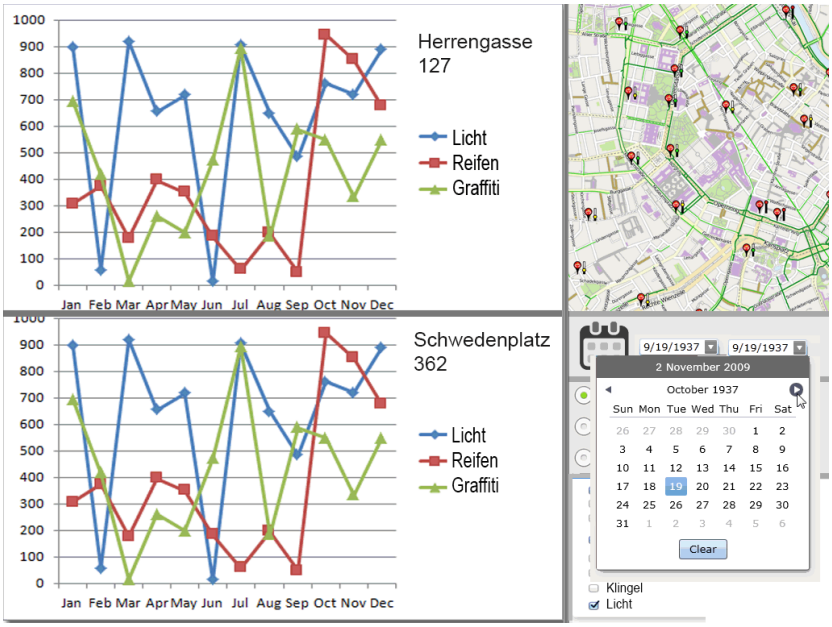
Approach

We started up developing some low fidelity prototypes, which we presented to the students in the visualization course. As a result of their feedback we tried to build a first high fidelity prototype, in which we combined several functionalities of our low fidelity prototypes. But before we’re getting there, we’d like to provide a closer look at our prototypes:

Prototype 1

This prototype provides different coloured glyphs. Our city bike customer can decide in this view to look at damage reports or driver frequency respectively. More views could be inserted here. In the maps container he can see the attribute's value in form of value sized circles. On the right navigation bar there will be a bar chart which will show more detailed information. E.g. if one clicks on damage reports the global size of them is seen in the maps whereas the bar graph shows which kind of damages occurred at picked station. So there is a closer view of those damage reports categorized through the different causes of the damage. The same option is available for the stations traffic workload and other inserted views if needed.

Prototype 2



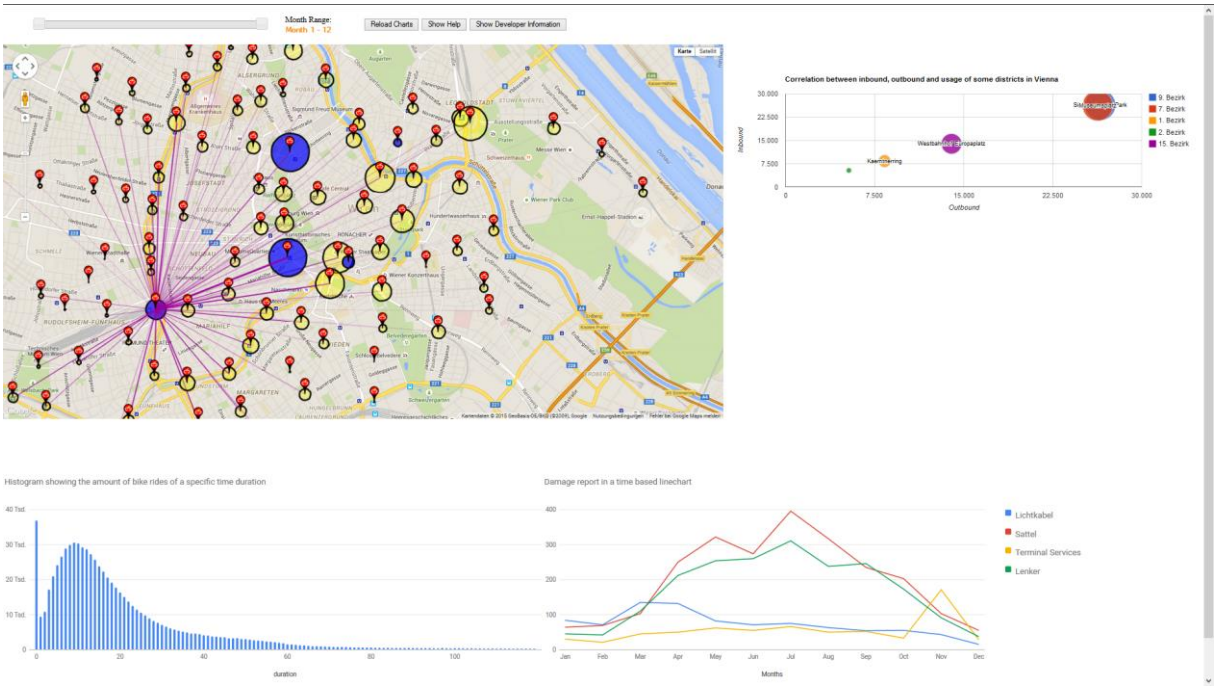
This prototype provides a station comparison on different parameters which the city bike customer himself can decide which one to pick. If the customer has two stations near him he would certainly like to

compare them so that he knows which one to aim. As a customer you are interested in the bike's condition and you like to go to the station which is less frequent used so you can certainly get a bike at every time. Parameters like damage reports or driven routes or driven-kilometres etc. can be chosen at the right navigation bar after picking the stations on the map in the right top corner. In the low right corner are details of the attributes to pick if possible.

Prototype 3

Let's assume the customer is interested in gathering statistic information about the service. With this prototype a report for how this service is used will be accomplished. With the knowledge of average kilometres per hour e.g. one can assume that around the station there is heavy traffic or other factors which reduce biking speed. Besides that the customer wants to get statistics over a period of time. The summer month will differ from the winter month in many aspects. How much slower the bikers go e.g. will be able to see.
The prototype provides the city bike customer a route-drawing visualization for each station by measuring the traffic between the different stations. The line-thickness and colour depend on the frequency usage. Furthermore there is a scatterplot which shows us the duration of the driving time on the x-axis and the driven kilometres on the y-axis. So you can say, this scatterplot shows us the station, with the fastest cyclists. There is also the possibility to filter those visualizations by month. In the end, there is also a bar chart, which could show for example the different damage reports.

Because the dataset limited us in some ways we basically changed most of our first solution attempts and created a visualization which combined some minor characteristics of the low fidelity prototypes, but inherently turned out to be a whole new mock up.



In the left top corner there is an interactive map which shows us all the citybike stations. If you click on a station, the lines between stations are drawn. These lines represent the trafficamount (represented through the line thickness) among the stations. The terminals coloured circles illustrate the total amount of traffic at the stations. If the circle is bigger there is more traffic, if the circle is smaller there is less traffic. The panel on the right, gives us some developer information about the dataset and will be replaced with the charts. Also it's possible to select more than one station, this is important for our comparable charts, which will be added at a later point. To select more than one station you have to press the "CRTL"-key and hold it while clicking on a station.

On the right side the bubblechart gets the station-ID from the clickable map, depending on which station is selected. It shows the correlation between incoming and outgoing rides, as well as the entire usage for the chosen stations. It is also possible to see the incoming and outgoing rides for specific months through the slider over the interactive map. This will also change the x- and y- axis's range. The color of the bubbles are dependent of their districts and the size of the entire usage. If two stations have the same color, then they're in the same district. Also there is a mouseover-effect to see a tooltip which holds the specific details.

In the left bottom corner we implemented a histogram. The bins of the histogram (on the x-axis) are the minutes spent on the bike per ride. The y-axis shows us the amount of rides with this specific loan duration. Like the bubblechart, the histogram is affected by the month-picker. So we can look at the driven time durations for a specific month or just for the summer month.

The last chart, placed under the bubblechart, is a line chart, which is also affected by the monthpicker and shows the annual progress of the damage-report amounts. With this view, we'd like to point out the most common damage reports in the city bike service and in which month they appear.

Implementation

The WebApplication includes JavaScript-Files, JavaScript Libraries, CSS Files, Data Files and Configuration Files.

Providing the data

Because the dataset we got in the beginning is quite distributed we had to think about a way to map the different datasets into a centralized and therefore easy usable datasource. We wrote a C#.NET tool to map the different dataset together, the tool loops through the files, maps them together and creates a JSON-File for the WebApplication. The JSONs hold information about stations, amount of rides and other stuff for example locations. Starting with several comprehensive datasets, it was quite hard to provide a small-sized source, that the runtime won't suffer from loading times. Here is a short look at a specific station:


```
[{
    "TERMINALID": "2002",
    "STATIONID": "1054",
    "STATION": "Jaegerstrasse U6",
    "LNG": "16.369837235391667",
    "LAT": "48.235129924432414",
    "BEZIRK": "20",
    "RIDESENTIRE": 13112,
    "RIDESOUTBOUND": 6548,
    "RIDESINCOMING": 6564,
    "INBOUNDJANUARY": 169,
    "INBOUNDFEBRUARY": 151,
    "INBOUNDMARCH": 499,
    "INBOUNDAPRIL": 547,
    "INBOUNDMAY": 900,
    "INBOUNDJUNE": 834,
    "INBOUNDJULY": 801,
    "INBOUNDAUGUST": 866,
    "INBOUNDSEPTEMBER": 764,
    "INBOUNDOCTOBER": 531,
    "INBOUNDNOVEMBER": 309,
    "INBOUNDDECEMBER": 140,
    "RIDES": [{
        "STATIONID": "1046",
        "ENTIRE": 302,
        "JANUARY": 18,
        "FEBRUARY": 5,
        "MARCH": 25,
        "APRIL": 41,
        "MAY": 40,
        "JUNE": 37,
        "JULY": 22,
        "AUGUST": 27,
        "SEPTEMBER": 37,
        "OCTOBER": 21,
        "NOVEMBER": 11,
        "DECEMBER": 18
```


Implementation details

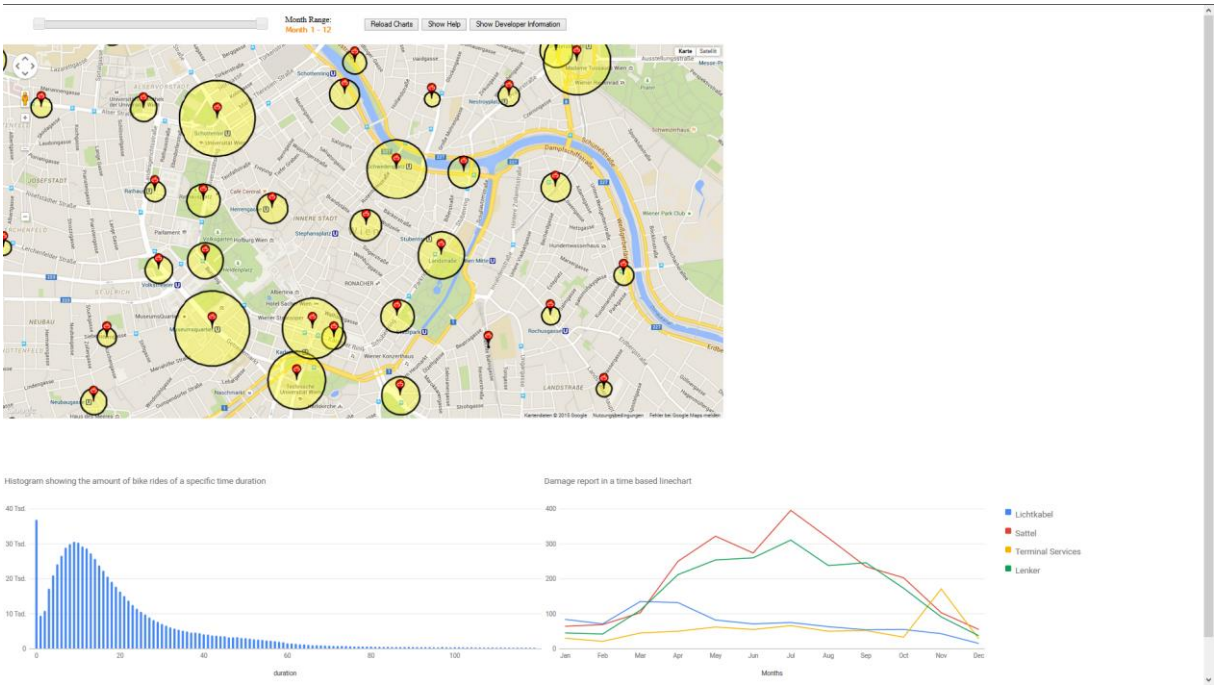
We have separate JavaScript-Files for the map and the charts. We tried to build the map and the charts as stand-alone modules to keep them exchangeable. We are using JavaScript Libraries like jQuery, google maps API, google charts API and d3 to accomplish our different tasks. Our data files are JSONs and

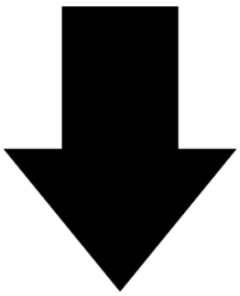
configuration XMLs. The XMLs are used by the WebApplication to load colours and other design information. Our idea was, to change parts of the design just in the XML without changing the code, it's faster and the error-rate keeps lower.

Results

Typical user scenario

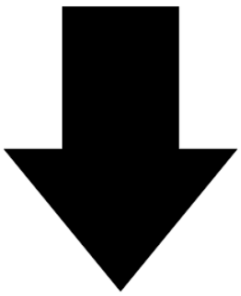
If you’re first loading the webapplication you’ll see this screen. There are no stations pre-selected, so there is no array availaible which could be passed to the bubble chart. The month picker is set to the whole year per default.





Now we’re clicking on a station

On the map you can see the driven routes. The line thickness represents the amount of connections between those two stations.



Now we're selecting a few stations with STRG + Clicking

Selected stations now turn blue and will be passed to the bubble chart. The bubble chart shows the correlation between the in- and outbound of the rides of a station. The colour will represent a specific district, which can be easily resolved by the legend on the right side.

In the next step we're changing the selected month-range

After changing the month range all of the charts got updated.

Every chart in this visualization is provided with a mouseover effect to get more specific details.

Performance of the system

The performance is not optimal. Every time when the slider is moved the program loops through the json file. This could be optimized by loading the json files to the client browser and then calculate the view on the local machine.

User Feedback

We asked 5 persons to try our app. We just told them it is a page about citybikes in Vienna and provided them with no further information. We just let them handle this tool and watched them. Just a 2 of them used the help button to understand the clicking option, the rest tried to click the stations, but where not provided with the bubble chart, because of the fact, that they didn't use CTRL+Click. All of them told us that the charts are very easy to interpret (After we told them that there is another chart if you know how to select multiple stations).
All of them found the app to be a little bit laggy when selecting a station from the map. One of those users made a wish for more active elements to make it more playful.
Two users criticized that the app is only stable in the Firefox browser. One of this two testpersons also wanted the charts on the bottom to change when you select a station, which is unfortunately not possible due to our dataset). And three users mentioned that the lines in the map are just not so easy to follow, because they sometimes overlap or they are just hard to connect to an endstation.

Discussion

One strength of our app is definitely the fact that it is coded in JavaScript. Therefore it is easy to extend/add new functionalities to the Webapp because of its modularity. Apps with tableau are easy to begin with but are limited in their power of extendibility and functionality.

A weakness our app has is the level of detail because of the dataset which limited us. The lower two graphs get the information globally and do not correlate with the selected stations. We would have liked to implement it but damage reports have no ID for stations. It is Browser dependent. It's not usable with Safari, but with Google Chrome and Mozilla Firefox. Whereas Firefox is slightly better than Chrome. Another weakness is, that the Vertical and horizontal axis of the Bubblechart and Linechart are not fixed. They move with the value of the objects.

Lessons learned

- Proper use of json objects
- How to handle data sets and put them in an easy useable form
- If it's hopeless to implement a specific idea, just let go of it and don't use most of the time working on that specific idea. Develop a different one.
- Use of Google map/chart API

References

https://developers.google.com/chart/interactive/docs/

https://en.wikipedia.org/wiki/Data-driven_journalism

Tamara Munzner, "Visualization Analysis and Design" 2014

http://stackoverflow.com/

http://json.org/

http://www.citybikewien.at/