

Web-based Visualization of Daily Mobility Patterns in R

Antje von Schmidt, Rita Cyganski, Matthias Heinrichs

Institute of Transport Research
German Aerospace Center (DLR)
Germany, Berlin

e-mail: antje.vonschmidt@dlr.de, rita.cyganski@dlr.de, matthias.heinrichs@dlr.de

Abstract-Human mobility reflects how, when, where and why people move from one location to another. Transport demand models can provide answers to these questions. Such models usually require a large amount of data as input and provide detailed information about the trips made by each individual during a day. Exploring this data can become very complex. Usually, several types of aggregation and disaggregation are performed on a spatial, temporal or demographic level. Often a variety of tools is used for analyzing, communicating and validating the data. This paper introduces an interactive and scalable web application for analyzing, communicating and validating the data of a transport model. The presented approach could also be used within a different research domain. Therefore, recommendations and implementation notes are given on how such an application can be realized in R. A special focus is set on the representation of daily mobility patterns within this application.

Keywords-transport demand; daily mobility patterns; visualization; R; Shiny.

I. INTRODUCTION

This paper is an extended version of previous work presented at the IARIA Seventh International Conference on Data Analytics [1].

Human mobility reflects how, when, where and why people move from one location to another. Biking, going by car, using public transport, and walking are often considered modes to accomplish activities such as education, working, shopping, leisure or other things. Transport models can be used to provide a realistic picture of the current traffic situation, to predict future developments of transport demand or to undertake scenario-based analysis of various possible development paths, such as an aging population, changed prices or new mobility trends. To estimate the demand that is analyzed and visualized in this paper, the microscopic transport demand model TAPAS [2][3] is used. Such models usually require a large amount of data as input. Within a simulation run, TAPAS calculates the activities and trips performed during a day for each person in the research area. Thereby, it provides individual trip chains with specific spatial and temporal information as well as a detailed description of each person and the associated household as simulation output. The sum of these trips results in an overall picture of the transport demand within a specified study area and timescale.

Not only the validation of input and output data, but also the analysis and representation of simulation results are not easy tasks, especially if the target audience is heterogeneous

and several output media have to be covered. There are many visualization types available, and which one to choose strongly depends on the research question and the domain of interest. In the field of transport research, several types of aggregation and disaggregation are performed, may they be on a spatial, temporal, or demographic level. A wide range of visualization tools are available [4]. Some of them can be used out of the box, and others require programming skills or are only intended for a certain type of visualization, spatial or non-spatial. Consequently, often a combination of these tools is used, also because commercial software solutions or eye catching animations are usually too expensive for being applied within research projects. In general, a flexible and extensible approach is preferable, which allows adaptation to the respective needs.

The aim of this paper is to introduce the application “Transport Visualizer” (TraVis). It is an interactive, user-friendly and scalable web-based solution for analyzing, communicating and validating the simulation data of a transport model, such as TAPAS. A special focus is set on the representation of daily mobility pattern within this application. TraVis is based on the programming language R, which is known as a flexible and powerful open source language. R is widely used in the scientific field for statistical computing, data analysis and visualization [5]. This application architecture was chosen because of the possibility of combining the benefits of R with the interactivity of modern user interfaces by using the Shiny web-framework [6].

The paper is organized as follows: Related work is discussed in Section II, followed by an introduction of the simulation data used with TAPAS in Section III. Section IV describes the implementation requirements and Section V gives recommendations, on how to set up a web-based application in R. The usage of spatial data and the realization of the TraVis application are outlined in Section VI respectively in Section VII. The visualizations of the daily mobility patterns within this application will be presented in Section VIII, using a baseline scenario for the city of Berlin in 2010. Finally, Section IX summarizes this paper and gives an outlook on future work.

II. RELATED WORK

For a long time, the role of visualization in transport modelling was largely limited to the static presentation of aggregated final results, primarily through statistical tables and simple graphs or the provision of key indicators describing the development of transport demand (see for

instance [7][8]). In recent years, map-based representations were progressively used within the transport domain. In particular, the level of detail is increasing: from a general overview of the study area, to a certain spatial resolution of it or temporal selections, down to the visualization of individual travel behavior [9]-[12]. Modern web technologies [13]-[15] as well as the growing availability of data and computing power have contributed to a significant increase in the usage of dynamic and interactive forms of visualization. In transport as in other research fields, it is necessary to analyze the visualization requirements and consider the type of data to be visualized, the audience to be addressed, the purpose of the visualization, the appropriate level of detail, the aspects of the data that should be transmitted, and the target medium for which the visualization is generated. A general framework for the visualization of transport data and models is defined in [16].

Another important factor is the type of visualization. Charts are often used to represent information on a high or intermediate level of detail. Which type of illustration should be used depends on the data type and on what is to be shown. In [17], a chart selector guide and four basic methods of data analysis are defined that can help to choose the right chart type for comparison, composition, distribution, and relation. The comparison of single values, such as totals or means, is best shown with regular bar or column charts, while line charts are more suitable for identifying distributions of continuous values or the development of a measure over time. Stacked charts can be used to represent the absolute or relative changes within the composition of categorical variables. Scatter and bubble charts are suitable for representing correlations between two respective three numerical variables, whereas parallel coordinates can be used to point out relationships between multivariate data. A chord diagram is a common way to illustrate interrelation between data in a matrix, whereas data with a spatial context are typically represented by suitable maps. For example, a choropleth map can help to identify regional varieties, while flow maps show the quantity of movements between geographical units. Spatial changes between scenarios are usually achieved by difference maps, whereas animated maps are suitable to represent differences over time. There are many more ways for visualizing data, and not all of them can be listed here. Hence, only a brief insight into the diversity is given. A good overview can be found at [18].

Although many visualization concepts and tools are available, the challenge remaining is to integrate these different approaches and to enable the users to perform their

analyses without mastering programming languages or using a variety of tools.

III. SIMULATION DATA

Microscopic transport demand models usually require a variety of different input data, such as population, locations, network, transport offers, timetable and land use. All this data is very heterogeneous in terms of format, spatial resolution, and time frame. TAPAS, for instance, requires all possible locations within a specified study area where activities such as working or shopping can be realized, but also the respective capacities indicating the number of people that can be there at the same time. Furthermore, a main input is a highly differentiated synthetic population, which is generated by the SYNTHESIZER [19]. Within such transport models, individual and household information play an important role in the choice of destination and transport means. Therefore, a synthetic population contains information both at the personal and household level. In the case of TAPAS, age, sex and an employment status variable are used on the individual level. In addition, information on available mobility options is required, such as a driving license, a public transport ticket, a bike as well as the budget a person can spend on mobility. Household information comprises the number of persons, the total household income, the number and type of vehicles that belong to the household as well as the spatial reference of the address. The simulation results, on the other hand, provide detailed information for each individual trip, including, among other, start time and duration, transport mode chosen, the distance as well as information about the trip purpose, and the location of origin respectively destination. Table I provides an excerpt of the simulation output. The identifier of each person and the corresponding household can be used to merge further information. Overall, data associated with an average simulation run is quite vast, reaching roughly 11.5 million trips using a baseline scenario for the city of Berlin with a population of 3.3 million inhabitants in 2010.

IV. IMPLEMENTATION REQUIREMENTS

Exploring the simulation data of a transport model can become very complex. The tabular representation of the simulation output, shown in Table I, is not easily to interpret by humans, especially when handling large amounts of data. Based on related work, the following aspects were selected as requirements for the implementation.

TABLE I. GENERATED INDIVIDUAL ACTIVITY BASED DAILY MOBILITY PLAN FOR A SAMPLE PERSON

P-ID	HH-ID	Trip				Purpose			Location	
		Start time	Duration	Mode	Distance	Activity	Start time	Duration	Start	End
1	5	7:15	10 min	bike	2 km	shopping	7:25	5 min	at home	at a shop
1	5	7:30	30 min	public transport	10 km	working	8:00	8:30 h	at a shop	at work
1	5	16:30	15 min	bike	3 km	leisure	16:45	1:00 h	at work	at a gym
1	5	17:45	45 min	bike	9 km	leisure	18:30	12:00 h	at a gym	at home

A. Data type

As mentioned before, the simulation data includes a variety of heterogeneous data types, which all have to be taken into account for exploring the data and the resulting daily mobility pattern in a disaggregated or aggregated manner. Besides the data with a spatial context such as activity location, traffic flow, population density or origin-destination matrices, there are also descriptive variables. These can be further subdivided into continuous, discrete and categorical data, such as the trip length, person age, or the activity performed.

B. Target audience and purpose

The visualizer shall help modelers to review the model values and disseminate the results to a wider audience, including both the scientific community and the public. Therefore, the usage of different media, including scientific publications, static presentations or interactive visualizations is intended.

C. Level of detail

The target application should contain all levels of detail of spatial, temporal, and demographic dimensions. It should be possible to aggregate simulated data at different viewing modes – for the complete study area or parts of it. This can be used, e.g., to validate the applied synthetic population, including the vehicle fleet and mobility options. The simulation result should be used for computing common key indicators of the transport demand (e.g., modal split) by aggregation. At the most detailed level, the individual travel behavior should be extractable from the simulation output and visualized. In addition, the usage of public space should be included in the analysis. It might be interesting to know how much space is used for parking vehicles. This is particularly interesting due to the space limitations and competition in cities. The space requirements for parking may change, for example through an increasing use of mobility services (e.g., bike-, car- or scooter-sharing) or the introduction of autonomous vehicles. Therefore, this information can provide useful information for future urban planning.

D. Output medium and interactivity

According to the different audience and purpose, both static as well as dynamic media have to be addressed. To understand changes, for example between different scenario parameters or time frames, it is important to provide the possibility to compare the corresponding simulation results. To take a closer look at certain aspects of the simulation output, the use of filters should be supported, including the following filter types: specific groups of persons or households, mode choice, location, trip purpose, time of traveling, distance, trip type and specific part of the study area. Detailed filter options are shown in Table II. These filters should be used to limit the simulation result accordingly.

TABLE II. FILTER OPTIONS

Filter	Elements
Region type	agglomeration, urban, sub urban, rural, zone identifier
Household size	1, 2, 3, 4, 5+
Number of cars	0, 1, 2
Person group	kids (< 6), pupil, student, employed, unemployed, pensioner
Transport mode	walk, bicycle, car, car (co-driver), public transport, other
Activities	education, leisure, private matters, shopping, working, other
Locations	education, leisure, private matters, shopping, work, other
Trip type	local people, commuters, origin, destination
Time of traveling	early (0 - 6), morning (7- 12), afternoon (13 - 18), evening (19 - 24)
Distance	1000m, 2000m, 3000m, 4000m, 5000m, 5000m+

Another main aspect is the opportunity to adjust the spatial scale, whereby different resolution of zones or the European wide standardization for geographical grids [20] may be applied. To address the various target media, it is also required to export appropriate figures for print media and to show the results on screen.

E. Visualization type and customization

With regards to the different types of data and the various target media, the application should contain suitable visualizations, such as different charts, maps and animations. This is necessary for presenting the simulation data according to the desired level of detail. Furthermore, the application should be scalable so that new types of reporting can be integrated immediately. Besides the choice of the right visualization type, it is also very important to have the opportunity to adopt its appearance. Therefore, it is required to customize attributes related to the layout, such as the color of the bars, the chart background, and the position of labels or of the legend as well as the appropriate font type and size. With the focus on internationalization, it is also necessary to supply different languages in order to automatically generate the required labels for the visualization.

V. RECOMMENDATIONS ON HOW TO SET UP AN WEB-BASED APPLICATION IN R

The development of a web-based application is not always straightforward. If a comprehensive application has to be created, the implementation can quickly become complex and time-consuming. It is therefore recommended to create readable and reusable code. In the following section, some recommendations are given, on how to set up a web-based application in R.

A. Use a development environment

RStudio, for instance, has established its position as a development environment in the R community. This tool should be used in conjunction with an up to date R version. Both are platform independent and a freely accessible version is available. When creating a new application, select "Shiny Web Application" as the project type within RStudio. This automatically integrates the Shiny web-framework. After the creation, a first example application is available and can be adapted accordingly.

B. Apply a usable application structure

A basic Shiny application is usually structured either in one script (*app.R*) or split into two: a user-interface script (*ui.R*) and a computational script (*server.R*). The *ui.R* script includes the layout and all the user-interface elements of the application, whereas the logic is specified in the *server.R* script. Here, for example, is defined what will occur if the user changes the value of a select box. Both scripts interact with each other through input and output objects. Splitting the programming code in only two scripts may be sufficient for smaller applications, but the programming code can quickly become unwieldy in larger ones. Therefore, it is recommended to split these main scripts further to make the code easier to maintain for developers. This can be additionally supported by a modular application design.

C. Create modules to use elements several times

Recently, the capability of using modules was added to Shiny as a new feature [21]. Shiny modules can be used to capture functionality and avoid name collisions by using namespaces for the input and output elements. This is especially important as element identifiers must be unique within a Shiny application. A module can be considered as a function that can be called several times within an application and is also structured either in one script (*module.R*) or split into two: a user-interface script (*ui.R*) and a computational script (*server.R*). To use modules across applications, it is recommended to pack them into a separate R package. This provides the full benefits of packaging, such as an update of the module package will automatically update all affected Shiny applications accordingly. Before creating modules, one should consider what recurring features are needed in the application.

D. Define the data store used

Data is often stored in a database. PostgreSQL, for example, can be used as a database system. A free version is also available. With the PostGIS extension, spatial data can also be managed within such a database. But, data can also be stored in text files instead of in a database. Within R it is possible to access all common data types within various data stores.

E. Reduce the data volume

Within the R environment, all data is stored in the working memory. When analyzing large amounts of data, this can slow down the performance. Therefore it is

recommended to reduce the data volume. This can be done, for example, by aggregating the data accordingly.

F. Create a concept for managing spatial data

Spatial related evaluations within the field of transport research have to be often carried out for different spatial scales. Furthermore, the boundaries of a spatial zone may change or new zonal levels are added. Therefore, the data have often been spatially adapted accordingly. The spatial mapping of data on the fly can be very time consuming. For this reason, it is recommended to create a concept to avoid data redundancies and an unnecessarily inflation of the data volume.

G. Use HTML widgets for interactive visualization types

A Shiny application can easily be extended by including HTML widgets. They provide an interface to a specific JavaScript library and generate interactive visualizations. The usage within R usually does not require further web development skills. There are already several widgets for interactive visualization types available, such as for charts, maps, or data tables.

H. Provide multi-language support

Results of research projects often have to be presented on national and international conferences or in journals. Therefore, it is necessary to provide the labels of the figures in the respective language. The labels should therefore not be integrated statically into the application, but instead be dynamically adaptable. This can be achieved, for example, by providing and using resource files for each language.



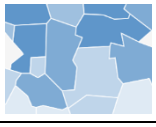
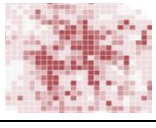
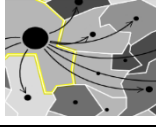
I. Decide, how to share the application with others

A Shiny application can be started locally within RStudio. To make this application available, the entire application folder with the associated files can be passed to other users. To start this application, it is necessary that the user has RStudio and R installed on the computer. This type of application sharing may be sufficient within a workgroup. However, it is also possible to run the application on an own web server. Therefore, the Shiny Server can be installed on a Linux operating system. This server is freely available as open source. In addition, such an application can also be hosted for free or commercially via the RStudio service.

VI. USING SPATIAL DATA IN R

Spatial data plays an important role within the simulation data. Not only for the preparation of the input data, but also in the later analysis and visualization of the simulations. Table III includes possible geometry types to represent transport related issues. Whereby, relevant attributes of the types can be used to highlight content related issues. The color of a point for example can be used to present different activity locations, whereas the size can be used to visualize the location capacity and the line thickness can help to show a different traffic volume.

TABLE III. GEOMETRY TYPES TO REPRESENT TRANSPORT RELATED ISSUES

Geometry type		Intended purpose (e.g.)
Point		activity location, location capacity
Line		origin and destination of a trip
Polygon		population density
Grid		car density
Combination		aggregated traffic flow

This kind of data may be available in various formats, e.g., as shapefile, text file or in a database. R offers several packages for handling spatial data. The package *rgdal* [22] for instance includes functions for reading and writing shapefiles. Data available in the geoJSON format can be integrated using the *geojsonio* package [23]. The *rpostgis* package [24] provides an interface to PostGIS enabled PostgreSQL databases. Currently, spatial objects in R are often based on spatial classes which are specified in the *sp* package [25]. The recently created *sf* package [26] supports simple features, a standardized way to encode spatial vector data and could replace the *sp* package in the future. The package links to a lot of other R packages, include many operations to manipulate spatial data, and can be used for reading and writing the mentioned formats. Furthermore, the *sf* package interacts very nicely with the *tidyverse* package [27] and makes it quite easy to create maps in conjunction with other appropriate packages.

Depending on the intended use, maps may be needed either in a static or interactive way. For instance, the packages *cartography* [28], *ggplot2* [29] or *tmap* [30] can be used to plot static maps, whereas interactive maps can be created with packages such as *leaflet* [31], *highcharter* [32] or as well with *tmap*.

VII. REALIZATION OF THE TRAVIS APPLIACTION

Based on the implementation requirements, the following parts of the application were realized as described below.

A. Concept for managing spatial data within a database

All simulation data is stored in a PostGIS enabled PostgreSQL database. The database structure of TAPAS is designed to avoid redundant data and unnecessary increase in data volume. Therefore, each spatial scale of a study area is kept in its own table, including an identifier, the coordinates of the boundaries as polygon and further attributes like the zone name. In addition, all household and location addresses as well as other places without direct address, such as playgrounds or parks, are stored as point coordinates in the *geocodes* table. All geometry columns of these tables have been provided with a spatial index. The use of spatial indexes can reduce the time required for spatial queries compared to tables without such an index. Every point coordinate within the *geocode* table is mapped to the desired spatial scales. Finally, the corresponding zone identifier has been stored within the *geocodes* table, each scale in its own column. These identifiers can be used to join the data from the belonging *spatial scale* table. Each point coordinate has also a unique identifier which is joined to the *households* and *locations* table. At the same time, the household and location identifier is included in the *trips* table. With this approach, the spatial mapping of the point coordinates to the polygon of the respective zone only has to be performed once. It therefore offers a great advantage for the later spatial analysis and visualization of the data, as the aggregation to a certain zone is much faster using identifiers than a spatial reallocation. Furthermore, new spatial scales can be integrated easily. An outline of the database structure is shown in Figure 1.

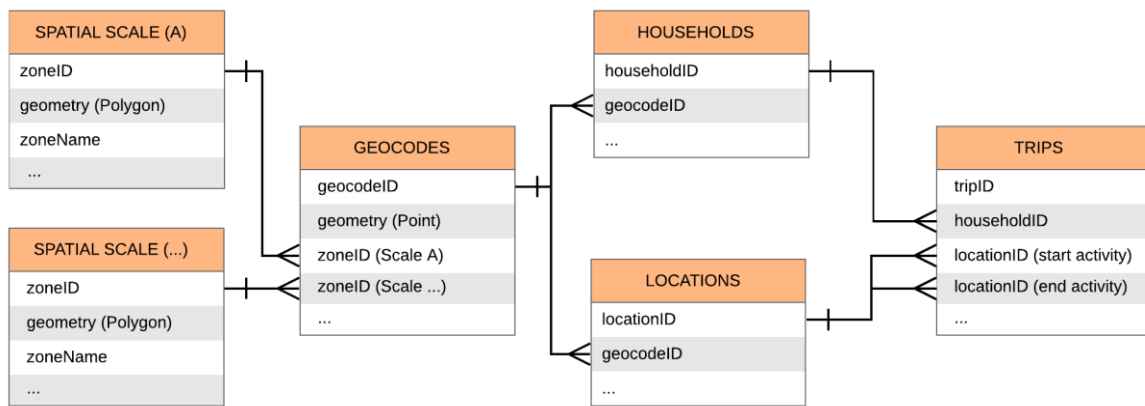


Figure 1. Outline of the TAPAS database structure

B. Approach to reduce the data volume

Loading a full sample, with roughly 11.5 million trips for the city of Berlin 2010, is not really manageable within R. Especially the representation within a web browser does not work well, because the loading of the data is rather slow. To reduce the amount of data, the disaggregated simulation data is not loaded directly into R. Instead, several SQL functions for data preparation were implemented within the database e.g., for data filtering, or data aggregation. This also makes parsing the data with other programs possible. However, the goal is to enable users to perform their analyses automatically and without the use of additional tools.

The simulation results generated by TAPAS are not overwritten when simulation parameters are adjusted; new result data is generated instead. Once generated, the data is static and can be stored in corresponding R objects. This has the advantage that the data does not have to be retrieved from the database again for a new simulation evaluation.

C. Application structure

The two main scripts of a basic Shiny application are split into several parts to make the code easier to maintain. Global objects for example, are defined once in the *global.R* file and used multiple times within other files. Application settings such as the used packages, property files for each provided language, or database configurations are defined within the settings folder. Each part of the application frontend (e.g., header, body, or sidebar) is placed in its own file and referenced within the *ui.R* script. Computational parts are outsourced as functions and used within the *server.R* script. Each created module is stored in its own folder, which contains the corresponding *ui.R* and *server.R* files of the module. Once generated, data (e.g., evaluated data) is stored within the data folder. The developed and used file structure within TraVis is shown in Table IV. An overview of the application architecture is given in Figure 2.

TABLE IV. USED FILE STRUCTURE

Folder/File	Description
/data	stored spatial and evaluated data (*.rdata)
/functions	outsourced functions including: database connection, text formatter
/modules	includes subfolder for each module, which contains the corresponding <i>ui.R</i> and <i>server.R</i> files
/server	computational snippets
/settings	application settings, e.g. used packages, properties (de/en), database config
/ui	user-interface snippets
/www	Stylesheets, JavaScript functions
global.R	global objects, with reference to /settings and /scripts
server.R	main computational file, with reference to /server and /scripts/modules
ui.R	main user-interface file, with reference to /ui and /scripts/modules

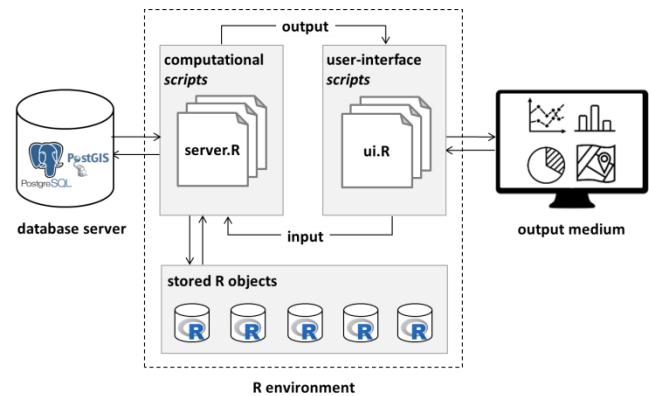


Figure 2. Overview of the TraVis application architecture

D. Modularizing

The following recurring features were defined for the TraVis application so far:

- Render a map view, enabling support for panning, zooming and switching layer on/off, as well as for exporting the map as a printable image. Supply a settings panel to adjust the map layout individually. This includes customization options, such as the color scheme and legend position.
- Supply a map data panel with elements to switch between multiple scenarios and possible categorical values.
- Render a chart view and supply a settings panel to adjust the chart layout individually. This includes customization options, such as the type of the chart, an adoptable color scheme for matching the respective project's corporate design, text alignment and legend position. Furthermore, provide the possibility to export the generated chart as a printable image.
- Render a panel with value boxes to highlight key indicators.
- Supply a filter panel with elements to limit the simulation result according to the filter options given in Table II.
- Enable an interactive link between the above mentioned elements.

For flexibility reasons and further usage within other applications, these defined features have been split into separate Shiny modules: *map*, *chart*, *box* and *filter* module. In the following, each of the sub-modules is described in more detail.

The user-interface of the *map* module contains a data panel with selection fields for the choice of the respective simulation and the associated categorical data. Furthermore, there is a map container and a corresponding settings panel for adapting the map layout. The server part of the *map* module receives several parameters as input, such as spatial aggregated data and the associated geometry. Additionally, it provides a function to view the data according to the selected scenario as well as the chosen category (e.g., the mode

walking). For rendering the map, the *leaflet* HTML widget [31], related to the JavaScript *Leaflet* library is integrated within this module. This widget is perfect for displaying maps within a web browser, as all the interactive features, such as panning, zooming and switching layer on/off, come into use. The module structure, including the definition and use of the namespace as well as the linkage of the input and output objects between the user- and server part of the *map* module is outlined in Figure 3.

The *chart* module includes a chart container and a corresponding settings panel for adapting the chart layout within the user-interface. The server part receives several parameters as input, such as aggregated data and the colors to use as default. For rendering the chart, the *highcharter* HTML widget [32], related to the JavaScript *Highcharts* library is used within this module. *Highcharts* contains a large number of interactive chart types and supports the export into different output formats (e.g., png, svg or jpeg). This module includes additionally a function to render the chart according to the selected spatial object within the map.

The *box* module contains a panel with several boxes in the user-interface. The server part, on the other hand, receives the values as parameter and contains a function for rendering these boxes. For this purpose, the value boxes from the *shinydashboardPlus* [33] are used. Additionally, this module contains a function to render the boxes according to the selected spatial object within the map.

The user-interface of the *filter* module contains a panel with select boxes for each defined filter option in Table II. The server part contains a function to render this panel and to filter the data accordingly. Finally, this module is used to return the filtered data to a parent module.

```
#definition of the map module user-interface
mapUI <- function(id){

#namespace definition
ns <- NS(id)
...
#use of the namespace for the data panel elements
uiOutput(ns("simulation")) #list of simulations
uiOutput(ns("category")) #list of modes, activities, ...
...

#map container
leafletOutput(ns("map_leaflet"), ...)
...
}

#definition of the map module server part
map <- function(input, output, session, data, geometry, named.list, ...){

ns <- session$ns
...
#render ui-elements for the data panel
output$simulation <- renderUI({... selectInput(ns("simulation")) ...})
output$category <- renderUI({... selectInput(ns("category")) ...})
...
#render map
output$map_leaflet<- renderLeaflet({...})
}
```

Figure 3. Outline of the *map* module

The *map*, *chart*, *box* and *filter* module are wrapped within a higher-level *simOut* module. This module includes a grid layout for the arrangement of the four sub-modules and a reference to the respective user-interface elements of each module. The server part of the *simOut* module receives several parameters such as spatial aggregated data, the associated geometry and a named list. Some of these inputs are provided as parameters for the sub-modules, others are used within function. The named list, for example, is used for transforming categorical values into adequate factors. Furthermore, it includes a function to aggregate data as well as the reference to each sub-module. The *simOut* module has been created for the evaluation of the simulation output. An outline of the *simOut* module is shown in Figure 4. It illustrates the link between the four sub-modules and the *simOut* module. Furthermore, a *simIn* module has been created for the evaluation of the simulation input data. Both main modules are used several times within the application.

```
#module user-interface
simOutUI <- function(id){

#namespace definition
ns <- NS(id)
...
#insert the user-interface parts of the corresponding modules
filterUI(ns("filter"))
mapUI(ns("map"))
boxUI(ns("box"))
chartUI(ns("chart"))
...
}

#module server
simOut <- function(input, output, session, data, geometry, named.list, ...)
...
#call the server parts of the corresponding modules
callModule(filter, "filter", ...)
callModule(map, "map", ...)
callModule(box, "box", ...)
callModule(chart, "chart", ...)
}
```

Figure 4. Outline of the parent module *simOut*

E. Application frontend

The application frontend of TraVis is built with an extended dashboard version for Shiny applications, *shinydashboardPlus* [33]. It includes four parts: header, body, left and right sidebar. The header is currently used for displaying the application name and to toggle both sidebars. Further elements can be integrated. The left sidebar contains the main navigation of the application. Each menu item refers to an evaluation topic. Currently, parts of the population, vehicle fleet and stationary traffic from the simulation input can be visualized as well as daily mobility patterns which are obtained from the simulation output. The right sidebar is used as the main settings panel with several tabs. The first one enables the user to get access to the data. Here, it is possible to select one or more simulation runs belonging to a chosen research project. The spatial scale can be defined in the second tab. The third tab allows for

defining the target language to be used throughout the entire session. Following the approach of multi-language support within a Shiny application [34], switching between English and German is currently implemented. The body includes the main panel. Depending on the chosen evaluation topic, the corresponding module *simIn* respective *simOut* is used.

VIII. DAILY MOBILITY PATTERNS FOR BERLIN

Daily mobility patterns can be described by three key indicators that are related to the selected mode of out-of-home travelers [35]: the number of trips, the distance and the travel time. Table V contains common daily mobility key indicators using a baseline scenario for the city of Berlin in 2010. Approximately 3.3 million inhabitants lived in 1.9 million households that year.

The result of a microscopic transport model allows working with even more detailed patterns of travel behavior. For instance, the daily mobility can be influenced by various socio-demographic factors such as age, gender and income. Especially, the interaction between the trip purpose and the used mode can be analyzed. But, also the spatiotemporal information is very important because these values can be used to point out spatial differences in the traffic volume during a day and to visualize traffic flows. The following figures illustrate various forms of representing daily mobility patterns within TraVis. The focus is on the visualization of the computed traffic volume grouped by transport mode for the city of Berlin in 2010. Within the application it is possible to visualize the spatial distribution for different spatial scales and for each transport mode as well as the activity made.

TABLE V. KEY INDICATORS OF DAILY MOBILITY USING A BASELINE SCENARIO FOR THE CITY OF BERLIN IN 2010

Mode	Number of Trips	Distance (km)	Time (min)
	<i>Sum Values</i>	<i>Average Values</i>	
Bike	1.469.023 (12.5%)	5.1	29
Car	3.625.876 (30.9%)	7.6	25
Public transport	3.210.767 (27.4%)	7.8	50
Walk	3.422.100 (29.2%)	1.8	26
<i>Total</i>	11.727.766	5.9	31

Figure 5 (A) shows the spatial distribution of all walking trips within the city. The total share of walking trips for the entire city is shown in Figure 5 (B). The modal split of two selected districts is represented in Figure 5 (C). These districts, one within the city center and the second further out, serve to compare the traffic volume during a day. While, the traffic volume in the morning is shown at the top and in the afternoon at the bottom for each district, see Figure 6 (A-B). In both districts the morning pick takes place at 6 and 7 o'clock. Throughout the day there are more trips within the inner city district which are made by walking respectively by using public transport. In the further out district, however, going by car dominates. In addition, it is evident that in the afternoon many more trips are undertaken within the inner city district.

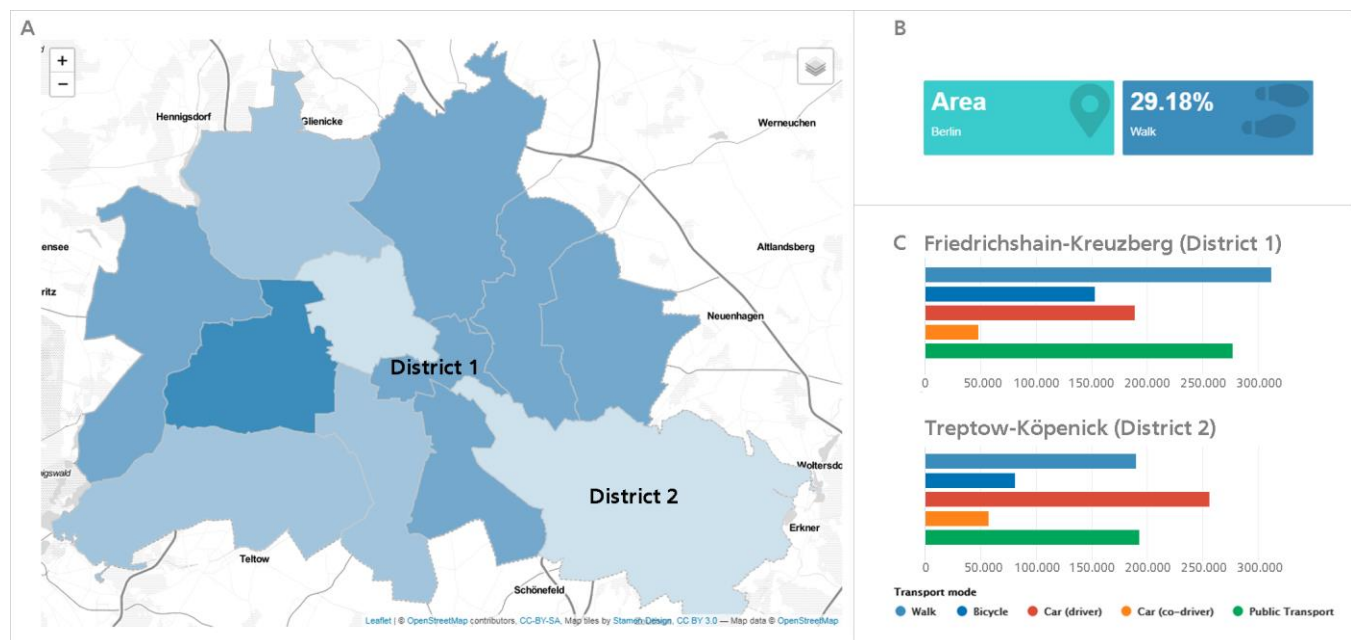


Figure 5. Visualization of the computed traffic volume for a selected mode of transport (walk). The spatial distribution within the city is shown in the map on the left side (A), while (B) represents the total share of walking trips for the entire city of Berlin. The modal split of two selected districts, one within the city center (Friedrichshain-Kreuzberg) and one further out (Treptow-Köpenick), is shown on the right-hand side (C).

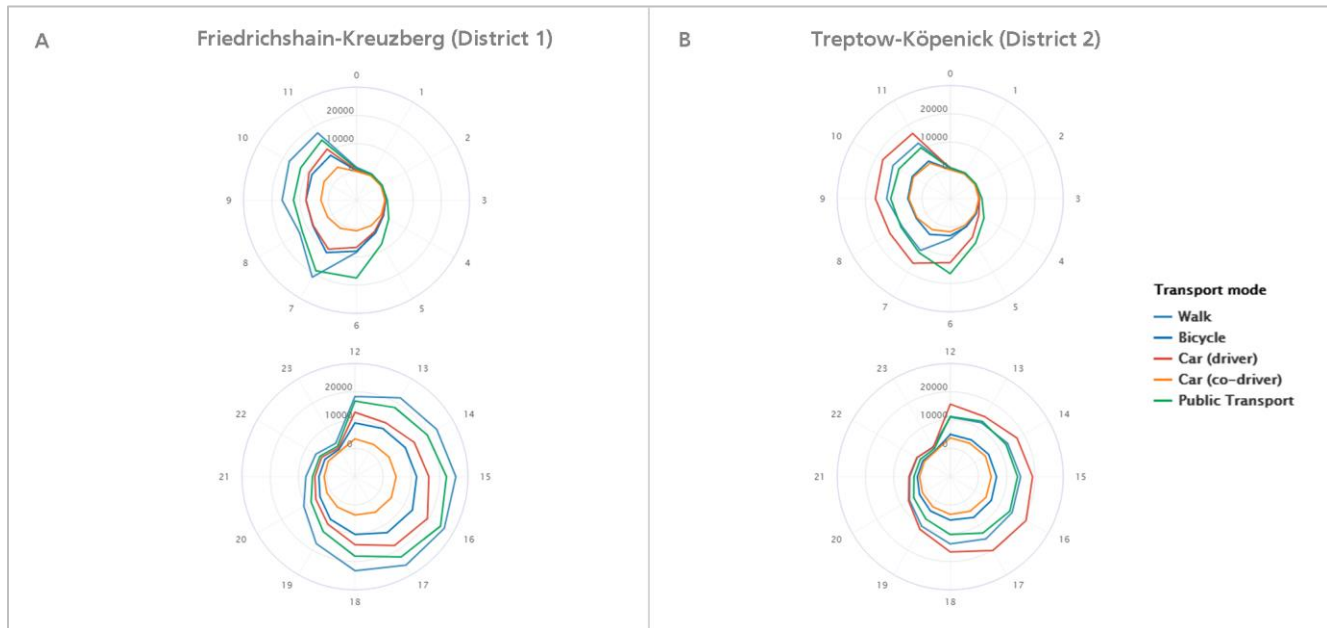


Figure 6. Visualization of the computed traffic volume during a day. Two districts within the city of Berlin are compared, one within the city center (A) and the second further out (B). The traffic volume in the morning is shown at the top and in the afternoon at the bottom.

IX. SUMMARY AND FUTURE WORK

Transport demand models can provide answers to the questions of how, when, where and why people move from one location to another. Exploring the data of such a model can become very complex. This paper introduces the web-based application TraVis, which integrates several types of aggregation and disaggregation on a spatial, temporal, or demographic level. The application is implemented in the programming language R, which is widely used in the scientific field for statistical computing, data analysis and visualization. By using the Shiny web-framework it is possible to convert R analyzes into an interactive web application. Therefore, the application architecture was chosen. TraVis is currently used to analyze and communicate the simulation data of the transport model TAPAS, but is also suitable for evaluating real-time mobility observation data. In this case, legal issues such as data privacy related aspects have to be considered.

The presented approach could also be used to analyze research results within a different domain. Therefore, several recommendations and implementation notes are provided on how such application can be realized in R. This includes the use of a development environment, a usable application structure, a modular application design, the use of a reduced data volume, a concept for managing spatial data, to integrate HTML widgets for interactive visualization types, the support of multi-language, and advice on how to share the application with other users. Finally, the visualization of daily mobility patterns within the TraVis application is presented for a baseline scenario for the city of Berlin in 2010.

Upcoming work will focus on the visualization of spatiotemporal related data such as the individual travel

trajectory and on implementing the ability to toggle between single and multi-window view. Furthermore, it is planned to make the application freely available as open source. TraVis is currently intended for internal use only, but it will be accessible from the following repository in the near future: <https://github.com/DLR-VF/TraVis>.

REFERENCES

- [1] A. von Schmidt, R. Cyganski, and M. Heinrichs, "Shine on Transport Model Simulation Data: Web-based Visualization in R using Shiny", in *DATA ANALYTICS 2018, The Seventh International Conference on Data Analytics*, pp. 67-72, 2018, ISBN 978-1-61208-681-1, ISSN 2308-4464.
- [2] M. Heinrichs, D. Krajzewicz, R. Cyganski, and A. von Schmidt, "Disaggregated car fleets in microscopic travel demand modelling", *7th International Conference on Ambient Systems, Networks and Technologies*, pp. 155-162, 2016, doi: 10.1016/j.procs.2016.04.111.
- [3] M. Heinrichs, D. Krajzewicz, R. Cyganski, and A. von Schmidt, "Introduction of car sharing into existing car fleets in microscopic travel demand modelling", *Personal and Ubiquitous Computing*, Springer, pp. 1055-1065, 2017 doi: 10.1007/s00779-017-1031-3.
- [4] N. Yau, "Visualize this: the flowingdata guide to design, visualization, and statistics", Wiley Publishing, 2011, ISBN: 978-0-470-94488-2.
- [5] R Core Team, "R: A language and environment for statistical computing", R Foundation for Statistical Computing, Vienna, Austria, 2018. <http://www.R-project.org>, accessed: 2019.11.26
- [6] W. Chang, J. Cheng, J. J. Allaire, Y. Xie, and J. McPherson, "Shiny: Web Application Framework for R", 2017. <https://CRAN.R-project.org/package=shiny>, accessed: 2019.11.26
- [7] J. L. Bowman, M. A. Bradley, and J. Gibb, "The Sacramento Activity-based Travel Demand Model: Estimation and Validation Results", presented at the European Transport Conference, 2006.

- [8] R. M. Pendyala, R. Kitamura, A. Kikuchi, T. Yamamoto, and S. Fujii, "Florida Activity Mobility Simulator: Overview and Preliminary Validation Results", *Transportation Research Record* (1921), pp. 123-130, 2005.
- [9] X. Liu, W. Y. Yan, and J. Y. Chow, "Time-geographic relationships between vector fields of activity patterns and transport systems", in *Journal of Transport Geography*, 42, pp. 22-33, 2015.
- [10] O. Klein, "Visualizing Daily Mobility: Towards Other Modes of Representation", A. Banos and T. Thevenien (Eds.), *Geographical Information and Urban Transport Systems*, Wiley Online Library, pp. 167-220, 2013.
- [11] H. Guo, Z. Wang, B. Yu, H. Zhao, and X. Yuan, "Tripvista: Triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection", *Visualization Symposium (PacificVis)*, IEEE Pacific, pp. 163-170, 2011, ISBN: 978-1-61284-935-5.
- [12] R. Cyganski, A. von Schmidt, and D. Teske, "Applying Geovisualisation to Validate and Communicate Simulation Results of an Activity-based Travel Demand Model", *GI Forum*, in *Journal for Geographic Information Science*, pp. 575-578, 2015.
- [13] M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-Driven Documents", *IEEE Transactions on Visualization and Computer Graphics*, IEEE Press, 17 (12), pp. 2301-2309, 2011, doi: 10.1109/TVCG.2011.185.
- [14] V. Agafonkin, "Leaflet: a JavaScript library for interactive maps", 2011. Available from: <https://leafletjs.com>, accessed: 2019.11.26
- [15] Highsoft, "Highcharts: interactive JavaScript charts for your webpage", 2009. <https://www.highcharts.com>, accessed: 2019.11.26
- [16] M. Loidl et al., "GIS and Transport Modeling-Strengthening the Spatial Perspective", *ISPRS International Journal of Geo-Information*, 5(6), 2016, doi: 10.3390/ijgi5060084.
- [17] A. Abela, "Advanced Presentations by Design: Creating Communication That Drives Action", Pfeiffer, 2nd edition, 2013, ISBN: 978-1-118-34791-1.
- [18] Ferdio, "Data Viz Project". <http://datavizproject.com>, accessed: 2019.11.26
- [19] A. von Schmidt, R. Cyganski, and D. Krajzewicz, "Generation of synthetic populations for transport demand models, a comparison of methods taking Berlin as an example", "Generierung synthetischer Bevölkerungen für Verkehrsnachfragemodelle, ein Methodenvergleich am Beispiel von Berlin" (original title), In *HEUREKA'17 - Optimierung in Verkehr und Transport*, FGSV-Verlag, pp. 193-210, 2017.
- [20] INSPIRE, <http://inspire.ec.europa.eu>, accessed: 2019.11.26
- [21] J. Cheng, "Modularizing Shiny app code", 2017. <https://shiny.rstudio.com/articles/modules.html>, accessed: 2019.11.26
- [22] R. Bivand, T. Keitt, and B. Rowlingson, "rgdal: Bindings for the 'Geospatial' Data Abstraction Library", R package version 1.4-4, 2019. <https://CRAN.R-project.org/package=rgdal>, accessed: 2019.11.26
- [23] S. Chamberlain and A. Teucher, "geojsonio: Convert Data from and to 'GeoJSON' or 'TopoJSON'", R package version 0.7.0, 2019, <https://CRAN.R-project.org/package=geojsonio>, accessed: 2019.11.26
- [24] D. Bucklin and M. Basille, "rpostgis: linking R with a PostGIS spatial database", in *The R Journal*, 2018, 10(1), pp. 251-268. <https://journal.r-project.org/archive/2018/RJ-2018-025/index.html>, accessed: 2019.11.26
- [25] E. Pebesma and R. Bivand, "Classes and methods for spatial data in R", in *R News*, 5 (2), pp. 9 - 13, 2005.
- [26] E. Pebesma, "Simple Features for R: Standardized Support for Spatial Vector Data", in *The R Journal*, 10 (1), pp. 439-446, 2018, doi: 10.32614/RJ-2018-009.
- [27] H. Wickham, "tidyverse: Easily Install and Load the 'Tidyverse'", R package version 1.2.1, 2017. <https://CRAN.R-project.org/package=tidyverse>, accessed: 2019.11.26
- [28] T. Giraud and N. Lambert, "cartography: Create and Integrate Maps in your R Workflow", in *JOSS*, 1(4), 2016, doi: 10.21105/joss.00054.
- [29] H. Wickham, "ggplot2: Elegant Graphics for Data Analysis", Springer-Verlag New York, 2016.
- [30] M. Tennekes, "tmap: Thematic Maps in R", in *Journal of Statistical Software*, 84(6), pp. 1-39, 2018, doi: 10.18637/jss.v084.i06.
- [31] J. Cheng, B. Karambelkar, and Y. Xie, "leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library", R package version 2.0.2, 2018. <https://CRAN.R-project.org/package=leaflet>, accessed: 2019.11.26
- [32] J. Kunst, "highcharter: A Wrapper for the 'Highcharts' Library", 2019. <https://github.com/jbkunst/highcharter>, accessed: 2019.11.26
- [33] D. Granjon, "shinydashboardPlus: Add More 'AdminLTE2' Components to 'shinydashboard'", R package, 2019. <https://CRAN.R-project.org/package=shinydashboardPlus>, accessed: 2019.11.26
- [34] C. Ladroue, "Multilingual Shiny App", 2014. <https://github.com/chrislad/multilingualShinyApp>, accessed: 2019.11.26
- [35] S. Schönefelder and K. W. Axhausen, "Urban Rhythms and Travel Behaviour: Spatial and Temporal Phenomena of Daily Travel", *Transport and Society*, Ashgate, 2010, ISBN: 978-0-7546-7515-0.