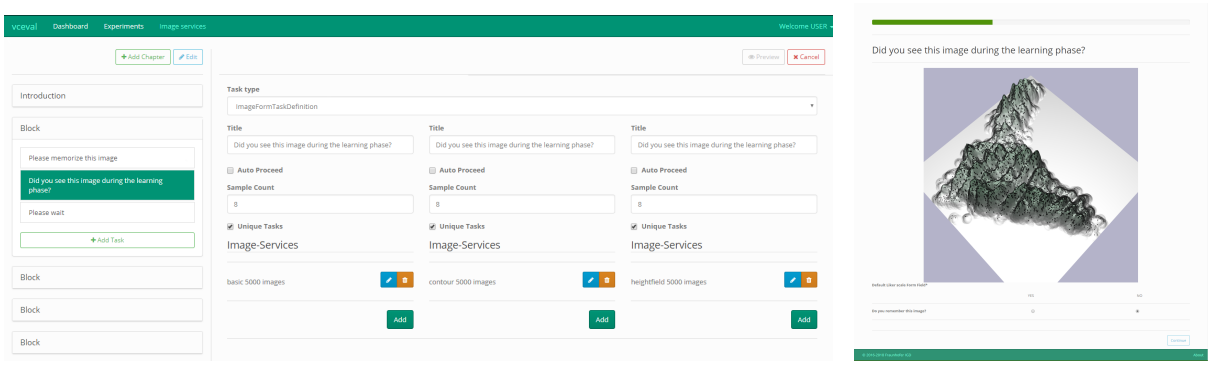
Lowering the Barrier for Successful Replication and Evaluation

Hendrik Lücke-Tieke* Fraunhofer IGD
Marcel Beuth† Fraunhofer IGD
Philipp Schader‡ Fraunhofer IGD
Thorsten May§ Fraunhofer IGD
Jürgen Bernard¶ TU Darmstadt
Jörn Kohlhammer‖ Fraunhofer IGD, TU Darmstadt



(a) Editing an between-group task

(b) The task as seen by participants

Figure 1: On the left, the experimenter configures an Image Form Task as part of a between-group chapter with three groups. As configured on the left, participants will repeat this task 8 times with different visualizations. On the right, a participant works on one task.

Abstract

Evaluation of a visualization technique is complex and time-consuming. We present a system that aims at easing design, creation and execution of controlled experiments for visualizations in the web. We include of parameterizable visualization generation services, thus separating the visualization implementation from study design and execution. This enables experimenters to design and run multiple experiments on the same visualization service in parallel, replicate experiments, and compare different visualization services quickly. The system supports the range from simple questionnaires to visualization-specific interaction techniques as well as automated task generation based on dynamic sampling of parameter spaces. We feature two examples to demonstrate our service-based approach. One example demonstrates how a suite of successive experiments can be conducted, while the other example includes an extended replication study.

Index Terms: Human-centered computing—Human computer interaction—User studies; Human-centered computing—Human computer interaction—Visualization design and evaluation methods; General and reference—Cross-computing tools and techniques—Empirical studies;

*e-mail: hendrik.luecke-tieke@igd.fraunhofer.de
†e-mail: marcel.beuth@igd.fraunhofer.de
‡e-mail: philipp.schader@igd.fraunhofer.de
§e-mail: thorsten.may@igd.fraunhofer.de
¶e-mail: juergen.bernard@gris.tu-darmstadt.de
‖e-mail: joern.kohlhammer@igd.fraunhofer.de

1 Introduction

Many scientists in the field of information visualization seek to produce valuable results. They strive to invent visualizations that help actual people solve real problems faster. Empirical studies are a rigid way to support the claim that a visualization technique achieves these high level goals. However, good quantitative evaluation requires a lot of effort [17, 19, 22], excluding any implementation-related efforts.

Even with maximum effort invested into an experiment setup, the most influential factor cannot be controlled. This factor is the experimenter – her explicit and tacit knowledge, assumptions, incentives, goals, and research culture. The PRIMAD model, developed in a Dagstuhl working group by Freire, Fuhr, and Rauber, included actors as variables that can be changed [12]. In this sense, a replication study is a variation of the variable "experimenter" in an experiment with a larger scope. While evaluation is a rigid method to substantiate actual scientific statements, only replication can confirm the validity of the original statement. However, the expectable merits do not always match the required investment. Even well documented laboratory experiments (with data, software and experiment protocol available in detail) still requires considerable effort to re-implement the experiment infrastructure and setup.

This leaves us with two basic strategies for fostering replication: Increasing incentives or lowering the required investment. Increasing incentives might have a stronger effect, but requires nothing less than cultural change. We therefore resort to the other strategy: The main goal of the application described later is to reduce the effort to actually set up, replicate and extend controlled experiments. In terms of Munzners nested model we focus on supporting visualization designers to validate visual encoding/interaction idiom approaches [25]. Within the categorization scheme of Isenberg et al., we target Qualitative Result Inspections (QRI) and User Performance (UP) tasks [17].

Our contribution is a web-based evaluation environment, tailored to conduct experiments with web-based visualization techniques.

Visualizations and their control parameters can be registered to this environment as a generic image generation service, hence referred to as "image service" in the remainder of this document. The authoring component of the tool helps building the experimental setup. An experimenter may freely combine task and visualization parameters. In addition, we support task randomization, task-order randomization, and subject-group based randomization (for within-group and between-group designs). Experimental results, including participant data and timings are automatically collected. The visualization services, experiment setup, and the results remain available for documentation and future (re-)use. We do not claim to be able to replicate every lab-study ever conducted. Still, a constrained evaluation environment might have a larger value for replication purposes.

We believe that the main benefit of our approach is the removal of important barriers for successful evaluation and replication of controlled experiments. For a first-time experiment, the design can easily be edited, tested, and improved over multiple iterations before leaving the pilot stage, thus avoiding the need to start from scratch over and over again. Replication benefits from this approach in particular, as the environment allows to repeat any experiment, as well as running variations of an original protocol (by cloning and adapting the parameters accordingly). This avoids translation losses while reinterpreting the protocol of an earlier study. We lower the barrier for evaluation of visualization techniques by reducing the effort required to include visualization-centric tasks into evaluation protocols. We remedy the necessity of manual stimulus generation by including a parameterization system into our system which automatically samples the parameter space to generate stimuli. The technical separation between experiment design and visualization technique also offers important benefits. It avoids adapting the experiment to the (proposed) solution, instead of adapting the experiment to the task. Additionally, it provides a lever to foster comparative studies of different visualizations and quickly test new hypotheses, as image services can be re-used.

The remainder of this paper is structured as follows: Section 2 outlines the current state of the art. We then describe our system in Section 3. To clarify the relevance, we show in Section 4, how experiment suites and replication studies can be set up. Finally, we conclude by discussing current limitations and future directions in Section 5.

2 Related work

We build our approach upon two complementary perspectives on evaluation in information visualization. First, we review works that focus on replication and reproducibility of evaluations. The second part surveys evaluation and survey systems, ranging from the scientific to the applied perspective.

2.1 Replication and Reproducibility

Donoho argues that the computer science community should adopt the model of the biostatistics journal, who introduced a reproducible research policy, which marks whether a published finding is backed by data, code, or both and if they are freely available [8]. The policy of the biostatistics journal follows Peng's definition of reproducibility [26]. Reproducible work enables independent researchers to verify results, and, due to availability of data or/and code, to perform alternative analyses. If independent researchers are able to reproduce results with other data and/or different methods, then they successfully replicated prior work.

Plaisant proposed in 2004 that (data and task) repositories and benchmarks should improve, which would foster quantitative evaluation and the comparability between approaches [27]. Her call for community efforts has not been answered to date. As our system allows integration of parameterizable image services, we support this proposition and provide an indirect solution to this: The more services are integrated on such a platform, the more data sets, results and experimental designs will be available out of the box for fellow scientists, who can build on those for replication and/or extension.

According to Isenberg et al., the majority of visualization evaluations were Qualitative Result Inspections (QRI). Authors publishing a new rendering method or visual encoding either compared the new method against an older one, or they argued for the superiority if no competitor approach was presented. For many of such evaluations, it is clear that the respective authors did not validate the approach empirically, and therefore the results of such work can only be described as anecdotal evidence. But theoretically, both types of QRI can be easily converted to User Performance (UP) evaluations, given enough time and resources, or a system that supports such evaluations [20].

Borgo et al. find that many papers that did crowdsourced evaluations failed to provide sufficient details about the study, making reproduction or validation of the study difficult [5]. This also holds for controlled lab experiments [17]. Access to the raw and anonymized experiment data allows for a more detailed comparison of where results may diverge, as well as validation of analysis techniques [5].

2.2 Evaluation and Survey Systems

Many web applications are available for designing surveys and questionnaires. To name a few, there are *surveymonkey* [36], *google forms* [13], *typeform* [38], *surveygizmo* [34], *qualtrics* [28], *surveyjs* [35] and *limesurvey* [21]. Most of these (all but [13]) support branching logic, which is essential for conducting between-subject studies. By using branching logic it is possible for example to distribute participants into groups depending on their answers before they conduct the actual tasks. To our knowledge, this has never been used in any study within the scope of information visualization and thus induces more complexity to the design process than necessary for between-subject experiments. Most of the products are commercial or have reduced functionality for non-paying users, e.g. limiting participants [11, 34, 36], allowing only few questions [38], or restricting the designer to work with simple question types [28, 36]. In general, these products do not treat visualizations as first-class citizens and provide no means to integrate external services.

More science-oriented applications for generic experiment design are *WeXtor* [29], *EvalBench* [2], *TouchStone* [22] and *tatool* [40]. However, with the exception of *WeXtor* and *tatool*, these do not support running the experiment in the web natively [29, 40]. Other science-related experiment designers are supporting just a single type of task [14, 32], or are highly specialized to fulfill the requirements of a scientific domain [23]. Many of the generic experiment applications do not provide a user interface for creating an experiment and/or require a reasonable knowledge of the respective programming languages, APIs, or configuration file schemata [23, 24, 40], which is necessary for efficient study creation. STEIN is an interesting approach to evaluate the user experience and usability of full systems [4]. However, it is targeted to record interaction, and thus not capable to validate variants of a system, e.g. in a between-subject comparison.

Heer and Bostock reported on a replication study comparing lab evaluations and crowdsourcing evaluations [15]. In their opinion the advantages of crowdsourcing (sample size, less biased population, cost) may compensate the lack of control in the remote evaluation setting. The dynamic generation of tasks was one of the features they missed in their study, which we consider one of the core assets of our approach. Ahmed, Zheng, and Mueller presented one of the first approaches that used the computer to sweep the parameter space in order to generate tasks [1] for a serious game that has been developed to evaluate color blending perception. Close to our work is the publication by Englund, Kottravel, and Ropinski, who described their evaluation system for scientific visualization [9]. They support automated stimulus generation by providing linear parameter sweeping [9]. Their approach is tightly coupled to the

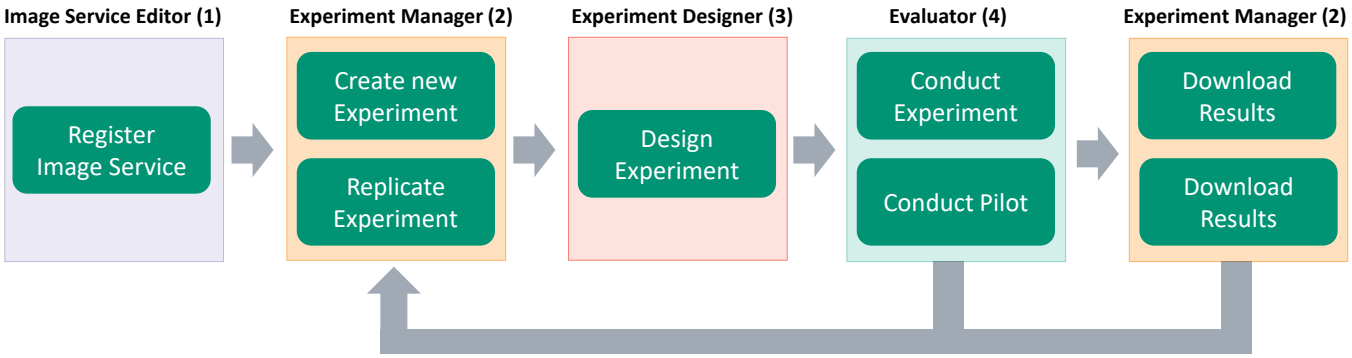


Figure 2: The prototypical work flow for experimenters. Registering image services is optional, designing experiments may be based on a previous study or designed from scratch. Generating stimuli is an automated process tied to each individual participant based on the experiment design. While conducting the experiment, progress can be monitored and (intermediate) results can be downloaded to prepare analysis.

scientific visualization framework InViWo [33]. In contrast, our system a employs loose coupling between visualization and test environment.

3 System Overview

Experimenters who want to evaluate a visualization technique, follow the work flow as depicted in Figure 2. They register a new image service in the **Image Service Editor** (Section 3.1), alongside the full parameter space it can handle. Then they use the **Experiment Manager** (Section 3.2) create (or clone) an experiment, which is then edited with the **Experiment Designer** (Section 3.3). Chapters and tasks are chosen, and where applicable, parameter space constraints (Section 3.3.3) are added. Afterwards, the experimenter uses the **Evaluator** (Section 3.4) to pilot the experiment, validate the correctness of recorded information, and eventually publishes it to conduct a study.

3.1 Image Service Editor

By choosing the approach of loose coupling between the actual stimulus generation and the evaluation system, we enable researchers to not only evaluate their own techniques, but to facilitate the comparison across different image services. Thus, we avoid implementing multiple stimulus generators by facilitating the re-use of existing ones.

3.1.1 Parameter Types

Currently, we distinguish between enumerable and continuous parameter types. The service definition covers the full parameter space, which can be restricted and constraint later on in the task definition. In practice, the **Enumerable Parameter Type** is more versatile, as it can represent categorical, ordinal, textual and numerical values. For example, a image service that generates a visualization of a node-link diagram using force-based layout may expose a parameter to modify the edge force calculation procedure using the identity function, the logarithmus naturalis, and the square root. Such a parameter might be registered using `edgeForceMod := [id, ln, sqrt]`. However, the external image service is required to be able to interpret these values correctly. Common use cases for this are discrete parameters like data transformation (e.g `dataset := [iris, cars, mnist]`), visual mappings (e.g. `colormap := [category10, divergingRedBlue, rainbowRbg]`) and view transformations (e.g. `colormap := [category10, divergingRedBlue, rainbowRbg]`). More examples are listed in the paragraphs on currently implemented services in Section 3.1.2.

For **Continuous Parameter Types**, we support natural numbers and real numbers. Both are defined by minimum (inclusive), maximum (inclusive), and discretization value, which defines the total number of equidistant stops between minimum and maximum. For example, a real-value range of `min := 0.0, max := 20.0, n := 9` results in a sampling space of $\{0.00, 2.22, \ldots, 17.78, 20.00\}$, while for natural numbers, the values round down to $\{0, 2, \ldots, 17, 20\}$.

To clarify, this defines the schema definition language for the parameters as it is used by the service editor. The actual parameters to be supported by an individual image service, depends on its implementation. Once defined, services may be used in the Experiment Designer to populate task definitions. Besides narrowing down the parameter space in the task definition it is possible to add constraints across image services. Both helps to reduce the degrees of freedom when it comes to sampling actual tasks. More details about configuring tasks can be found in Section 3.3.

3.1.2 Existing Image Services

In the following we show how actual service definitions may look like from a parameter definition perspective. These are based on examples from the later described replication study, already available image services are the following.

The **static image service** returns an image from a database based on the given name. We regularly use this service to introduce participants to a certain task type. Instead of actual stimuli, we for example use cat images, e.g we define `id := [cat1, cat2, cat3 ... cat12]`, after having registered 12 images of cats in the service interface itself.

The **number generator service** generates an image of a given number and color. This service has an integer parameter which reflects the number (`number := { min: 0, max: 9, n: 10 }`) and a enumerable parameter which corresponds to the color to be used (`color := [red, green, blue, yellow]`). This is a simple but helpful service for testing and piloting pre-designs, for example when a experiment is being set up *before* the actual target image service is ready for production use.

The **scatter plot service** is used in the replication of the Tory, Swindells, and Dreezer study [37] . The service supports three main render modes (dot-plot, contour-plot and 3D height fields) and a number of parameters from which some are exclusive to specific render modes. As our prototype system does not support exclusivity for parameters, we registered this service once for each of the three render modes. To illustrate the flexibility of the parameter types, and to show that they satisfy practically relevant use cases, we show the definition of each of those three services.

The **dot-plot mode scatter plot service** has the following configuration:

```
mode := [basic]
file := ["chainlink-100", "chainlink-500", "chainlink-
```

Table 1: Features in available products that are relevant to Information Visualization

Feature	This solution	SM[1]	GF[2]	TF[3]	LS[4]	SG[5]	QX[6]	SJ[7]
likert-scale questions	yes	yes	yes	yes	yes	yes	yes	yes
matrix values	yes	yes	yes	yes	yes	yes	yes	yes
multiple-choice	yes	yes	yes	yes	yes	yes	yes	yes
choose-one questions	yes	yes	yes	yes	yes	yes	yes	yes
dropdown selector		yes	yes	yes	yes	yes	yes	yes
free text	yes	yes	yes	yes	yes	yes	yes	yes
grouping task	yes					yes		
ranking/sorting task		yes		paid		yes	yes	no
text A/B test		paid	no				yes	no
chapters	yes	yes	yes	no	yes	yes	yes	yes
generate tasks from parameterized definition	yes	no	no	no	no	no	no	no
branching logic / conditions	simplified	yes	no	yes	yes	yes	yes	no
randomization of questions		yes				yes	yes	no
size limitations	no	# responses		# questions	# responses			
block randomisation	yes	paid		paid				

[1] survey monkey: `https://surveymonkey.com`
[2] google forms: `https://docs.google.com/forms`
[3] typeform: `https://typeform.com`
[4] lime survey: `https://limesurvey.org`
[5] survey gizmo: `https://surveygizmo.com`
[6] qualtrics: `https://www.qualtrics.com/`
[7] surveyjs: `https://surveyjs.io`

1000", "chainlink-5000", "chainlink-10000", ...]
backgroundColor := [#000000,#ffffff,#b4b3c9]
imageSize := { min: 128, max:2048, n: 1920}
kernelBandwidth := { min: 0.0, max: 100.0, n: 100000}
numColorSteps := { min: 2, max: 20, n: 18}
color := [#ff0000, #00ff00, #0000ff].

The **contour-plot mode variant scatter plot service** extends the dot-plot mode with one additional parameter, `drawPoints`, which had to registered alongside all parameters of the dot-plot mode variant. Additionally, the mode parameters was fixed to `contour`:
`mode := [contour]`
`drawPoints := [true, false]`.

And finally, the **3D height field scatter plot service** extends the dot-plot with the following parameters:
`mode := [parameters3D]`
`ambientLighting := { min: 0.01, max: 1.0, n: 100 }`
`cameraInclination: { min: 10, max: 80, n: 10 }`
`cameraAzimut := { min: 0, max: 345, n: 23 }`
`cameraDistance := { min: 0.1, max: 10.0, n: 100 }`
`cameraFov := [30,60,90,120]`
`heightFieldScaling := { min: 0.01, max: 20.0, n: 200 }`
`heightFieldSmoothingIterations := [0, 1, 2, 4, 8, 16, 32, 50, 64, 128]`.
Please note that `heightFieldSmoothingIterations` and some other parameters were defined as Enumerable Parameter Types, as this offers the freedom to constrain it to non-linear sampling. Although the services were registered as described, for the actual study, we fixed most of them to constant values, otherwise they would possibly introduce confounding factors.

3.2 Experiment Manager

The experimenter can create a new experiment from scratch, but also clone an existing experiment and adapt it accordingly. This is especially useful, when experiments are carried out in different stages of product development. As Kim, Yi, and Elmqvist already emphasized, piloting is important, and as many pilots as possible should be carried out [18]. Cloning experiments help exploring even ad-hoc variations of a visualization design in a convenient way, helping them to build suites of experiments. The decision to add this feature was born out of the motivation to replace "gut feeling" by as many experiments as needed.

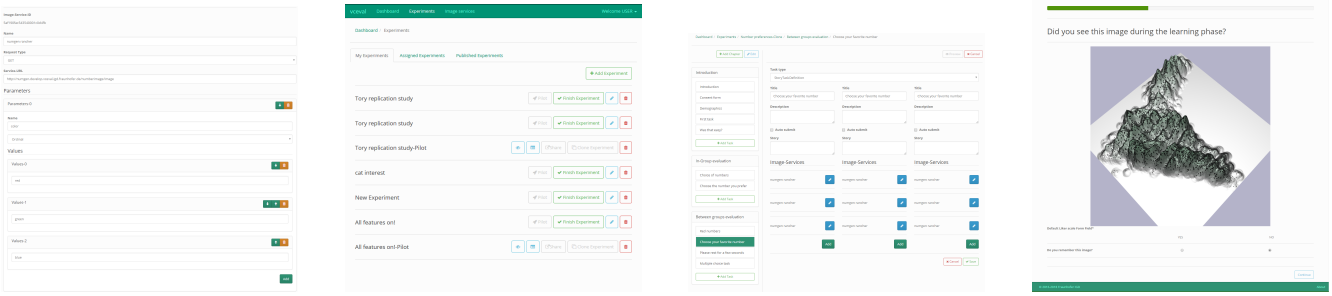
We agree with Donoho that the computer science community should incentivize reproducible research by marking publications which provide data, measurements and code freely [8]. To foster and ease publication of raw evaluation results, we therefore added the possibility to add links to external services and web sites. For example, the study results may be published on figshare [10] to publish results citeably, i.e. with an attached Digital Object Identifier (DOI), image services may refer to a homepage, e.g. github.com [7], an online version control system commonly used for source code hosting, for sharing details about an image service.

3.3 Experiment Designer

The Experiment Designer is the core of the application. Experimenters can configure and adapt an experiment according to their requirements. The main structural elements of an experiment are called Chapters. Each chapter represents a set of task definitions.

For each task, each chapter and the experiment itself, information such as title, description etc. can be defined. As randomization is a key requirement for many experiments, we include options to randomize within tasks, chapters and experiments.

Chapters Each chapter has a hypothesis, which helps the experimenter to focus on the specific hypothesis that should be tested. To test this hypothesis, the experimenter designs a "mini-experiment" within the larger context. As with any good experiment, participants should be introduced to the task ahead, as well as answer some example tasks to learn how to approach the actual tests. Chapters can contain many different tasks in any order. Also, between-group and within-group studies are defined on a per chapter basis. In all the survey and evaluation applications we examined prior this work, we realized that creating between-subject studies is mostly done by adding branching or show/hide logic to the sequence of tasks. This might be a good choice for classic surveys where e.g. the personal income should be requested only if the participant is employed,



(a) Image Service Editor (b) Experiment Manager (c) Experiment Designer (d) Evaluator

Figure 3: In the form (3a), a new image service is registered with name, service url and a single parameter 'color', but additional parameters can be added dynamically. Managing experiments, including cloning and downloading the results of published experiment can be done in (3b). In between-group chapters, tasks are shown next to each other to ease comparison between configurations (3c). Participants see a completely different user interface, in this case, they are asked if they remember the dynamically generated visual stimulus in form of a 3D height field (3d).

but it adds more complexity than needed. In most between-subject visualization studies different visualization types are presented to the groups. In such a case, it is easier for the researcher to see all variants of a task in parallel (see Figure 3c). To support within-group as well as between-group studies, the user has the option to specify the number of groups for a chapter. As soon as this number is changed, all tasks within this chapter are adjusted accordingly. To help configuring between-group tasks, all groups are displayed next to each other (see Figure 3c).

Tasks The experimenter can add tasks within a chapter. We distinguish between two types of tasks: those which contain image services (image tasks) and those which do not (generic tasks). The reason for this distinction is that: as image services provide a parameter space, this parameter space can be sampled dynamically to generate multiple instances of a task from the same definition (see also Section 3.4). Thus, the experimenter defines a task once for possibly hundreds of possible tasks that have to be solved by each participant.

3.3.1 Available Generic Task Types

Our main goal is to support experimenters in conducting evaluation on visualization techniques. But experimenters are free to select their target participants and therefore they can share the experiment on twitter, hire mechanical turks from crowdsourcing platforms, or even invite participants into their usability lab. As a result, in order to familiarize participants with their tasks even under remote working conditions, such a system requires general purpose tasks to be available.

Currently, the list of generic task types (i.e. non visualization related) listed below are sufficient to set up general purpose surveys:

Story Task The experimenter can define a so-called story in markdown, which is then displayed to the participant. It is suited for instructions, introductions, and a basic building block for any experiment (see fig. 4a). As with all tasks, time is tracked for story tasks as well, so this might be used as an indicator if a participant pays attention or not.

Pause Task As some experiments might take longer or short term memory needs "wiping", this task defines a break in the execution of an experiment.

Form Task The experimenter can add different form fields with a suitable label and description. It supports basic form fields like text field, number field, selection field, check-box field, date field, and image field. Also, some templates are built-in: a demographic information field group (gender, age, education, etc.), a participants consent group field, and a System Usability Scale (SUS) group [6]. Those templates were added to minimize the efforts required when requesting basic information. Last but not least, there is also a Likert-scale field where the experimenter can define the scale and the questions, and the participant will see a matrix with questions $\times$ scale. Basic form fields, templates and Likert-scale fields can be mixed freely, and the form will be appear in an concise form (shown in Figure 4b).

If used in conjunction with links to web applications, these task types can also be used to run usability or user experience studies on full systems, although this then requires multiple screens or windows open to conduct the study.

3.3.2 Available Image Task Types

However, to lower the efforts needed to evaluate visualization techniques, task types that are designed for the use with visualization techniques are necessary. Andrienko and Andrienko distinguish between elementary and synoptic tasks in visualization [3]. We enable experimenters to add synoptic tasks, e.g., outlier detection, target identification, and pattern/trend detection can be realized using an task for area or a lasso selection. Qualitative comparison, estimation and decision making can, to large extent, be recorded by using image form tasks, when adding appropriate questions to the form. Clustering can be realized with a grouping task.

The following list of currently implemented task types are explicitly designed to work on visualization techniques, e.g. the area selection task (see also Figure 4).

Multiple Choice Task One of the most important question types that can be used during an experiment is the comparison between different image services or different parameterizations of an image service. This comparison is possible with this task. The experimenter must define image services and their parameterization. During the execution of the experiment participants receive a list of images from which they must then select one image based on the given task.

Grouping Task We also added a Grouping Task which requires the participant to sort a given set of elements into predefined buckets. Again, the experimenter adds and parameterized multiple image services to the task and configures them.

Image Form Task In order to support gaining insight into the analytical reasoning process, the experimenter can add Image

Form Tasks. The participant is shown a image stimulus and has to answer questions (depicted in Figure 4e).

Lasso Selection Task As an experimenter you want to test in many applications if the participant is able to locate desired information in the visualization. One way to test this is to let the participant of an experiment mark a region in the visualization, depending on the task. It serves as an example that more complex interaction patterns can be integrated into our system (see Figure 4f). By using such interactions in an evaluation, more in-depth information can be acquired and therefore may provide hints on the reasoning process of the participant.

Area Selection Task For many tasks, a free shape provides too many degrees of freedom. If, for example, the participant is asked to locate the area with maximum density, a free-shape lasso selection makes the assessment very difficult. If the whole image is covered by the lasso, is that task solved correctly or not? Therefore, we added the Area Selection Task, which restricts the size of the selected rectangle. The extent of that rectangle can be defined by the experimenter.

These tasks can be divided into two groups: tasks displaying limited to one single image (Image Form Task, Lasso Selection Task, and Area Selection Task) and those allowing more (Multiple Choice Task and Grouping Task).

3.3.3 Controlling the Parameter Space per Task

Image tasks have in common, that they require (at least) one image service to operate on. As introduced in Section 3.3, the parameter space of an image service can be sampled dynamically to generate multiple instances of a task from the same definition. To enable fine-grained control over the visual stimulus generation, we added the possibility to reduce parameter spaces per task, and to add constraints between image services in the same task. For the Multiple Choice Task and the Grouping Task, which allow more than one image service, additional constraints are available to allow more sophisticated dependencies between image service parameterizations.

Task-based Parameter Space Reduction Different hypotheses require other image service parameters to be fixed, or reduced. To enable this, we added the possibility to reduce the parameter space for each image service on a per task basis. This reduced parameter space is then used in the sampling process that generates sequences of tasks for participants. The contour-plot variant of the scatter plot service has a binary parameter `drawPoints := [true, false]`. In the original study, all visual stimuli were rendering with points on the surface. Thus, to correctly replicate the original study protocol, the drawPoints parameter had to be fixed to true in all tasks.

Task-based Constraints on Image Services For Image Tasks with multiple image services, the experimenter can also add constraints. While parameter space reduction is important, it cannot capture more complex requirements. Constraints allow to model dependencies *between* image services and between different parameters of an image service. This can be useful if a visualization has for example two color parameters (e.g. background color and font color) and the experimenter wants to ensure that both differ. This is also helpful or if a task contains two image services and the number parameter between both should differ. Currently, we support a Unique Constraint and a Compare Constraint. Both work with Enumerable and Continuous parameter types. While the **Unique Constraint** ensures that the related parameters are equal or not, the **Compare Constraint** ensures an order relation between parameters, i.e. $<, =, >$.

3.4 Evaluator

While the Experiment Designer is used by researchers to specify an experiment, the Evaluator is used by participants to take part in a study. In other words, this provides the execution environment for conducting studies. The Evaluator has two functions: First, for each new participant of an experiment, it converts the experiment definition into a linear sequence of concrete tasks. Second, it tracks the participants progress through the experiment.

When new participants log into an published experiment, samples are dynamically generated for them (see Section 3.4). All defined chapters and tasks are parsed and mapped into tasks, adhering to all constraints that are defined. Tasks that contain a image services with parameterizations might result in multiple tasks, all other result in a single task. To generate a single parametrization, a sampling function can be applied for each parameter. In this approach, simple random sampling is used to draw from the defined parameter space, which assumes a uniform distribution among all values in that space. As a result, the participant proceeds through a linear series of concrete tasks. The different task types were already introduced in Section 3.3. In Figure 4, we give a visual overview of some the of tasks as seen by participants.

The Evaluator tracks answers given by the subjects, as well as the timing. Furthermore, it ensures that participants can proceed as configured, i.e. optional answers can be skipped and mandatory answers have to be recorded. The Evaluator also ensures that participants cannot skip through an experiment by manipulating HTTP requests as the back end keeps track of progress.

4 Examples

The first example evaluation is a suite of experiments that have been run with an early version of the presented system, which proofed the concept of loose coupling of image services.

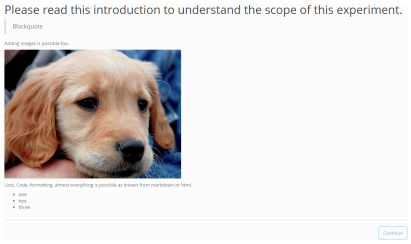
The second example is a replication study and highlights some benefits of our system. Our primary goal was to replicate Tory, Swindells, and Dreezer as the experimental design can be replicated quite easily [37]. However, the original study leaves open quite a few questions, which we then investigated, once the original study had been replicated.

4.1 Event Sequence Experiment I and II

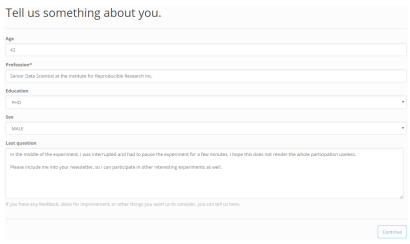
Often, an experiment and its results are just a step towards some greater goal. Therefore, it is important to put the experiment into context. This is what has been done with the event sequence experiments.

The first experiment was designed to find the best encoding for sequences of events. Position, color and shape were compared in a within-subject experiment. Per task, participants saw three event sequence visualizations: a baseline sequence, a target sequence and one distractor sequence. They had to decide whether the distractor or the target sequence was more similar to the baseline. The image service generated an image based on encoding (color, shape, position), allowed edit operations (insert, delete, replace), length of baseline sequence, edit distance to target and to distractor. The image service took care that the resulting images comply with the constraints. The results clearly indicated, that color is the most appropriate choice [30].

Based on that finding, a second experiment has been designed to study how the alignment of sequences supports or hinders participants in comparing event sequences that are encoded by color. A between-subject setup was chosen, i.e. one group had tasks that were setup just like in the predecessor (no alignment), one group had alignment via Needleham-Wunsch, and the third group had its sequences aligned by the longest common subsequence. The study had 50 participants with 180 measurements each. Quite a few significant effects were found, which will be presented in a different publication.

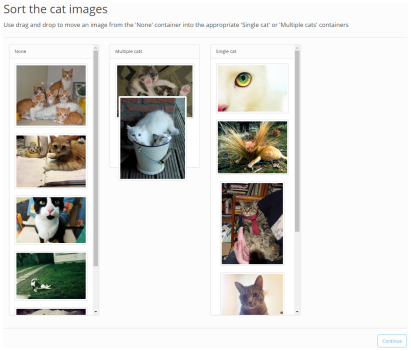


(a) Introductory texts and preparatory information can be presented in a formatted way

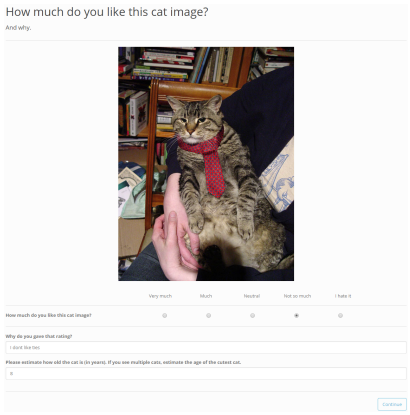


(b) For experimenter convenience, several form templates have been embedded, like this demographics form, which was enhanced with an additional question

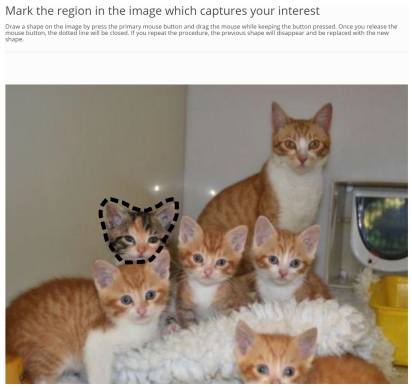
(c) Participants click on an image to directly make a choice. Here the participant selected the cat with the tie



(d) To group a set of images into categories, the user drags an image and drops it into the appropriate category



(e) To gain more information about the reasoning process of participants, a form can be combined with an image



(f) The Lasso Task lets participants select an arbitrarily shaped region of the presented image

Figure 4: Figure 4a and 4b showcase convenience tasks, Figure 4c and 4d provide a standard tasks for performance assessment tailored towards visualizations as first-class citizens. Figure 4e enables experimenters to gain insight into the reasoning process and fig. 4f emphasizes our focus on supporting visualization and analysis research.

4.2 Replication of an existing study

In "Comparing Dot and Landscape Spatializations for Visual Memory Differences", Tory, Swindells, and Dreezer conducted an experiment that compared three specialization designs for their ability to support mental operations involving visual memory. There are two reasons why we picked this study as the main example. First, it is well suited for replication as sufficient detail to derive the exact setup was presented. Merely the collection of stimuli has not been published. Secondly, we were genuinely interested if the study is the final answer to the question if 2D scatter plot really is best suited in all cases.

The original study was designed as a within-group design, with 30 participants. There were two conditions: 3 different visualization types and 2 data set sizes, i.e. 6 factors. Using 12 data sets, this resulted in 72 static images that served as stimuli for the participants. The experimenter took care that the images were easily distinguishable, i.e. the task itself was easy to solve. The order of stimuli was randomized to avoid learning effects. Each participant ran 6 tasks, one for each condition. The task was to learn 8 images (4 were targets, 4 were distractors). Then, participants were shown the 4 targets and 4 other distractors which they had to correctly label as "seen" or "new".

Prior to setting up the actual experiment, we derived different 2D distributions. We followed the advice of Henry and Fekete to modify real datasets or to generate datasets [16] and created some artificial and used some modified real world data sets (from the online repository OpenML [39]). These data sets were then saved in different sizes, where the smaller one always consists of a subset of the larger one. Applying the tagging scheme of Schulz et al., the data sets were generated using a hybrid approach based on imitation [31]. Once transferred to an image service which has been created to generate dot-maps, 2D landscapes and 3D landscapes from an input data set, we have set up the stimulus-providing side of the experiment.

We then set up the first version of the experiment using the Experiment Designer and ran a first pilot to (a) test the experimental setup and (b) discussed based on preflight pilots which data sets were appropriate and which were not. This sparked a discussion about the difficulty of tasks, and we concluded that it is worthwhile to investigate how much effect the data set size has. The altered experiment was piloted again, this time with 3 encodings and 4 different data set sizes. After discussing the experience again, we chose to split the groups, so that we diverged from the original within-group design and chose a between-group design. Each group has just one type of encoding, but all 4 data set sizes. Again, a small pilot was run and finally, the experiment was published and participants were invited.

Since we argue that it is easy to run suites of experiments, we report estimates for the creation of each experiment/pilot. Prior to creating the first pilot, the first version of the scatterplot renderer was implemented and registered (cf. Section 3.1.2), and data was acquired and prepared. Setting up the first pilot took approximately two days, including decisions on which service parameters to fix across all stimuli. During that period, the implementation of the

scatterplot renderer was continued, fully independent of the study design process. The second pilot was prepared in half a day (by cloning and adapting), while the third and last pilot took a full day, as we switched from within- to between-groups.

Our results with 58 participants can confirm the significant effect of the number of points on the accuracy. Due to our decision to split groups between image types, we have found no significant differences between image types yet. Statistical analysis was done by applying Repeated-Measures ANOVA. Furthermore, we measured overall accuracy of 86.7% (Tory: 87.1%) for dots, 83.4% (85.3%) for 3D landscapes, and 83.9% (79.1%) for 2D landscapes, indicating, that the difference across visualization techniques is not as large as indicated by Tory. However, we found that the time to answer probably correlates on the type of visualization ($p < 0.1$). Besides that, we could measure a 'fatigue' effect over the different chapters. Mean accuracy dropped for all image types over time, but the dots visualization was affected most. The full study is attached as supplemental material, which contains the experiment definition, all data sets and generated stimuli, as well as the anonymized results and the jupyter notebook that was used for the analysis.

5 Limitations and Future Work

The current version of the prototype does not integrate authentication and authorization, and is restricted to evaluation of static images. However, the support for **web-based interactive visualizations** is already planned. By adding a new kind of image service that produces embeddable web sites instead of images, the system can be extended without requiring a change in the process for the experimenter. Like static image services, interactive image services provide controllable parameters that will be configured in the Designer and instantiated in the Evaluator. To gather all relevant interaction information on the platform, we intend to incorporate participant activity tracing and collection approach described by Angelini et al. in their STEIN system [4].

Currently, support for **substantially different image services in the same task** is limited. This is due to the fact that each service has a different parameter space. To mitigate this, we will research, if a mapping functions between image services can be defined in such a way that constraints can be used too, or if the introduction of an domain specific language including variables is necessary.

Borgo et al. provided a preparatory checklist for crowdsourced experiments, which apply for most kinds of experiments [5]. We used their template to report the study results. Implementing this checklist – as far as possible – as automated checks during setup, is a viable next step towards the **establishing of a common format for evaluation**-related supplementary materials.

6 Conclusion

The pain points in creation of quantitative experiments that can be reduced with the help of our system are piloting, sampling, execution and data gathering. We chose a service-oriented approach which enables web visualization designers to plug new image generation services into the platform and conduct experiments. A recurring theme of the web application is re-use. Experimenters are able to re-use image services, to replicate existing studies, to re-run studies with variations. Many common task types, ranging from basic questionnaire tasks over elementary image-centric tasks to synoptic visualization-centered tasks are supported. Most implemented task types treat visualizations as a first class citizen. The separation of image service and execution environment supports in experimenters to designing and running multiple experiments on the same visualization service in parallel, replicate experiments and even compare against visualization services. We demonstrated the viability of our approach with two examples, one suite of successive experiments and one extended replication study.

Acknowledgments

The authors wish to thank Jan Burmeister for contributing the scatter plot rendering service which was used in the Tory replication study to generate stimuli. This work is partially funded by the Hessian Ministry of the Interior and Sports (HMdIS).

References

[1] Nafees Ahmed, Ziyi Zheng, and Klaus Mueller. "Human Computation in Visualization: Using Purpose Driven Games for Robust Evaluation of Visualization Algorithms". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (Dec. 2012), pp. 2104–2113. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.234.

[2] W. Aigner, S. Hoffmann, and A. Rind. "EvalBench: A Software Library for Visualization Evaluation". In: *Computer Graphics Forum* 32 (3pt1 June 2013), pp. 41–50. ISSN: 01677055. DOI: 10.1111/cgf.12091.

[3] Natalia Andrienko and Gennady Andrienko. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Berlin ; New York: Springer, 2006. 703 pp. ISBN: 978-3-540-25994-7.

[4] Marco Angelini et al. "STEIN: Speeding up Evaluation Activities With a Seamless Testing Environment INtegrator". In: The Eurographics Association, 2018. DOI: 10.2312/eurovisshort.20181083. URL: https://diglib.eg.org/handle/10.2312/eurovisshort20181083 (visited on 06/25/2018).

[5] Rita Borgo et al. "Information Visualization Evaluation Using Crowdsourcing". In: *Computer Graphics Forum* 37.3 (2018), pp. 573–595. DOI: 10.1111/cgf.13444.

[6] John Brooke. *SUS: A Quick and Dirty Usability Scale*. 1996.

[7] *Build Software Better, Together*. URL: https://github.com (visited on 06/28/2018).

[8] D. L. Donoho. "An Invitation to Reproducible Computational Research". In: 11.3 (July 1, 2010), pp. 385–388. ISSN: 1465-4644, 1468-4357. DOI: 10.1093/biostatistics/kxq028.

[9] R. Englund, S. Kottravel, and T. Ropinski. "A Crowdsourcing System for Integrated and Reproducible Evaluation in Scientific Visualization". In: *2016 IEEE Pacific Visualization Symposium (PacificVis)*. 2016 IEEE Pacific Visualization Symposium (PacificVis). Apr. 2016, pp. 40–47. DOI: 10.1109/PACIFICVIS.2016.7465249.

[10] *Figshare - Credit for All Your Research*. URL: https://figshare.com/ (visited on 06/28/2018).

[11] Holger Finger et al. "LabVanced: A Unified JavaScript Framework for Online Studies". International Conference on Computational Social Science (Cologne). 2017. URL: https://www.ic2s2.org/2017/elements/accepted_talks.html (visited on 06/22/2018).

[12] Juliana Freire, Norbert Fuhr, and Andreas Rauber. "Reproducibility of Data-Oriented Experiments in e-Science (Dagstuhl Seminar 16041)". In: *Dagstuhl Reports* 6.1 (2016). Ed. by Juliana Freire, Norbert Fuhr, and Andreas Rauber, pp. 108–159. ISSN: 2192-5283. DOI: 10.4230/DagRep.6.1.108.

[13] *Google Forms*. URL: https://docs.google.com/forms (visited on 06/22/2018).

[14] Yves Guiard et al. "Shakespeare's Complete Works As a Benchmark for Evaluating Multiscale Document Navigation Techniques". In: *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. BELIV '06. New York, NY, USA: ACM, 2006, pp. 1–6. ISBN: 978-1-59593-562-5. DOI: 10.1145/1168149.1168165.

[15] Jeffrey Heer and Michael Bostock. "Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. New York, NY, USA: ACM, 2010, pp. 203–212. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753357.

[16] Nathalie Henry and Jean-Daniel Fekete. "Evaluating Visual Table Data Understanding". In: *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. BELIV '06. New York, NY, USA: ACM, 2006, pp. 1–5. ISBN: 978-1-59593-562-5. DOI: 10.1145/1168149.1168154.

[17] T. Isenberg et al. "A Systematic Review on the Practice of Evaluating Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (Dec. 2013), pp. 2818–2827. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.126.

[18] Sung-Hee Kim, Ji Soo Yi, and Niklas Elmqvist. "Oopsy-Daisy: Failure Stories in Quantitative Evaluation Studies for Visualizations". In: *Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization, BELIV 2014, Paris, France, November 10, 2014*. Ed. by Heidi Lam et al. ACM, 2014, pp. 142–146. ISBN: 978-1-4503-3209-5. DOI: 10.1145/2669557.2669576.

[19] Robert Kosara and Caroline Ziemkiewicz. "Do Mechanical Turks Dream of Square Pie Charts?" In: *Proceedings of the 3rd BELIV'10 Workshop: BEyond Time and Errors: Novel evaLuation Methods for Information Visualization*. BELIV '10. New York, NY, USA: ACM, 2010, pp. 63–70. ISBN: 978-1-4503-0007-0. DOI: 10.1145/2110192.2110202.

[20] H. Lam et al. "Empirical Studies in Information Visualization: Seven Scenarios". In: *IEEE Transactions on Visualization and Computer Graphics* 18.9 (Sept. 2012), pp. 1520–1536. ISSN: 1077-2626. DOI: 10.1109/TVCG.2011.279.

[21] *LimeSurvey*. URL: https://www.limesurvey.org/ (visited on 06/22/2018).

[22] Wendy E. Mackay et al. "Touchstone: Exploratory Design of Experiments". In: ACM Press, 2007, p. 1425. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240840.

[23] Mario Kleiner, David Brainard, and Denis Pelli. "What's New in Psychtoolbox-3?" 30th European Conference on Visual Perception (Arezzo, Italy). 2007. URL: http://psychtoolbox.org/ (visited on 06/22/2018).

[24] Sebastiaan Mathôt, Daniel Schreij, and Jan Theeuwes. "OpenSesame: An Open-Source, Graphical Experiment Builder for the Social Sciences". In: *Behavior Research Methods* 44.2 (June 2012), pp. 314–324. ISSN: 1554-3528. DOI: 10.3758/s13428-011-0168-7.

[25] T. Munzner. "A Nested Model for Visualization Design and Validation". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (Nov. 2009), pp. 921–928. ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.111.

[26] Roger D. Peng. "Reproducible Research and Biostatistics". In: 10.3 (July 1, 2009), pp. 405–408. ISSN: 1465-4644. DOI: 10.1093/biostatistics/kxp014.

[27] Catherine Plaisant. "The Challenge of Information Visualization Evaluation". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '04. New York, NY, USA: ACM, 2004, pp. 109–116. ISBN: 978-1-58113-867-2. DOI: 10.1145/989863.989880.

[28] *Qualtrics*. URL: https://www.qualtrics.com/ (visited on 06/22/2018).

[29] Ulf-Dietrich Reips and Christoph Neuhaus. "WEXTOR: A Web-Based Tool for Generating and Visualizing Experimental Designs and Procedures". In: *Behavior Research Methods, Instruments, & Computers* 34.2 (May 2002), pp. 234–240. ISSN: 0743-3808, 1532-5970. DOI: 10.3758/BF03195449.

[30] R. A. Ruddle et al. *Methods and a Research Agenda for the Evaluation of Event Sequence Visualization Techniques*. Sept. 2016. URL: http://eventevent.github.io/papers/EVENT_2016_paper_9.pdf (visited on 09/09/2017).

[31] Christoph Schulz et al. "Generative Data Models for Validation and Evaluation of Visualization Techniques". In: *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*. BELIV '16. New York, NY, USA: ACM, 2016, pp. 112–124. ISBN: 978-1-4503-4818-8. DOI: 10.1145/2993901.2993907.

[32] R. William Soukoreff and I. Scott MacKenzie. "Generalized Fitts' Law Model Builder". In: ACM Press, 1995, pp. 113–114. ISBN: 978-0-89791-755-1. DOI: 10.1145/223355.223456.

[33] Erik Sunden et al. "Inviwo - An Extensible, Multi-Purpose Visualization Framework". In: IEEE, Oct. 2015, pp. 163–164. ISBN: 978-1-4673-9785-8. DOI: 10.1109/SciVis.2015.7429514.

[34] *SurveyGizmo*. URL: https://www.surveygizmo.com/ (visited on 06/22/2018).

[35] *SurveyJS*. URL: https://surveyjs.io/ (visited on 06/22/2018).

[36] *SurveyMonkey*. URL: https://www.surveymonkey.com/ (visited on 06/22/2018).

[37] Melanie Tory, Colin Swindells, and Rebecca Dreezer. "Comparing Dot and Landscape Spatializations for Visual Memory Differences". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1033–1040. ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.127.

[38] *Typeform*. URL: https://www.typeform.com/ (visited on 06/22/2018).

[39] Joaquin Vanschoren et al. "OpenML: Networked Science in Machine Learning". In: *ACM SIGKDD Explorations Newsletter* 15.2 (June 16, 2014), pp. 49–60. ISSN: 19310145. DOI: 10.1145/2641190.2641198.

[40] Claudia C. von Bastian, André Locher, and Michael Ruflin. "Tatool: A Java-Based Open-Source Programming Framework for Psychological Studies". In: *Behavior Research Methods* 45.1 (Mar. 2013), pp. 108–115. ISSN: 1554-3528. DOI: 10.3758/s13428-012-0224-y.