Psynteract: A flexible, cross-platform, open framework for interactive experiments

Felix Henninger^{1,3,4} · Pascal J. Kieslich^{2,3} · Benjamin E. Hilbig^{1,4}

Published online: 11 November 2016
© Psychonomic Society, Inc. 2016

Abstract We introduce a novel platform for interactive studies, that is, any form of study in which participants' experiences depend not only on their own responses, but also on those of other participants who complete the same study in parallel, for example a prisoner's dilemma or an ultimatum game. The software thus especially serves the rapidly growing field of strategic interaction research within psychology and behavioral economics. In contrast to all available software packages, our platform does not handle stimulus display and response collection itself. Instead, we provide a mechanism to extend existing experimental software to incorporate interactive functionality. This approach allows us to draw upon the capabilities already available, such as accuracy of temporal measurement, integration with auxiliary hardware such as eye-trackers or (neuro-)

physiological apparatus, and recent advances in experimental software, for example capturing response dynamics through mouse-tracking. Through integration with OpenSesame, an open-source graphical experiment builder, studies can be assembled via a drag-and-drop interface requiring little or no further programming skills. In addition, by using the same communication mechanism across software packages, we also enable interoperability between systems. Our source code, which provides support for all major operating systems and several popular experimental packages, can be freely used and distributed under an open source license. The communication protocols underlying its functionality are also well documented and easily adapted to further platforms. Code and documentation are available at <https://github.com/psynteract/>.

✉ Felix Henninger
mailbox@felixhenninger.com

Pascal J. Kieslich
kieslich@psychologie.uni-mannheim.de

Benjamin E. Hilbig
hilbig@uni-landau.de

¹ Cognitive Psychology Lab, University of Koblenz-Landau, Campus Landau, Building K, Fortstraße 7, D-76829 Landau, Germany

² Experimental Psychology, School of Social Sciences, University of Mannheim, Schloss Ehrenhof Ost, D-68131 Mannheim, Germany

³ Center for Doctoral Studies in Social and Behavioral Sciences, University of Mannheim, Mannheim, Germany

⁴ Max Planck Institute for Research on Collective Goods, Bonn, Germany

Keywords Strategic interaction · Economic games · Social dilemmas · Experimental design · Software · OpenSesame · Python · Process tracing

Introduction

Traditionally, participants in psychological studies have completed experiments in isolation, without affecting their fellow participants. Of course, this has been and is still hugely fruitful for researching a vast range of phenomena. However, many situations involve interaction and exchange between several individuals, such that outcomes and consequences depend not only on a single person's choice, but the combined behavior of several parties. Experiments that focus on individual participants' behavior cannot model these scenarios in their entirety.

As noted by Seithe, Morina, and Glöckner ([in press](#)), this interdependence of individual decisions is a feature of many

issues that face societies and humanity as a whole, where coordination and cooperation must be achieved in groups of agents with diverging interests. The most prominent example of such an interaction are social dilemma games (Van Lange, Joireman, Parks, & Van Dijk, 2013), in which an individual's self-interest and the collective gain are in direct conflict.

Pioneered by designs introduced in behavioral economics, a growing number of studies employ 'actual' interaction between participants and implement interdependence of participants as part of their design. That is, participants respond in parallel and incur consequences depending on their joint decisions rather than completing purely hypothetical scenarios or interacting with simulated agents. The motivation underlying this type of design is to capture the consequential nature of many interdependent decisions, and to introduce true conflict between response options with different ramifications for oneself and others, rather than inducing conflict between one's actual preference and the desire for impression management. Following this line of reasoning, there is a strong expectation particularly in economics to use incentivized, no-deception research and consequential, real-time interaction (Hertwig & Ortmann, 2001).

From the perspective of psychology, there is a growing interest in going beyond choices as the sole dependent variable, as these often provide only limited information regarding the psychological processes underlying interdependent decisions. To gain insight into these processes, more recent research has combined the paradigm of interactive decisions or social dilemmas with process measures. These measures include response times (e.g. Rand, Greene, & Nowak, 2012), eye-tracking as a measure of attention paid to different pieces of information (e.g. Fiedler, Glöckner, Nicklisch, & Dickert, 2013; Fiedler & Glöckner, 2015; Stewart, Gächter, Noguchi, & Mullett, 2016), or mouse-tracking as a proxy for the cognitive conflict experienced during the decision (Kieslich & Hilbig, 2014).

With these two perspectives come two distinct technical approaches to building experiments: On one hand, experimental software in psychology is typically general-purpose, extensible and flexible to accommodate the wide range of designs, methods, and dependent variables used (e.g. Mathôt, Schreij, & Theeuwes, 2012; Psychology Software Tools Inc., 2012). In addition, researchers can rely on the proven capabilities of these packages, such as stability and temporal accuracy. However, these software packages currently do not offer the option of creating interactive experiments: In the exemplary studies cited above that combine strategic interaction with somewhat more sophisticated data collection methods, the decisions observed were either hypothetical, outcomes were computed *ex post* rather than in real-time, or data were collected using home-grown

solutions involving large amounts of custom-built code. On the other hand, researchers building interactive designs with a focus on choices can draw upon several specialized software packages built exclusively for this purpose (cf. Janssen, Lee, & Waring, 2014), most prominently z-Tree (Fischbacher, 2007), and more recently BoXS (Seithe et al., *in press*) and oTree (Chen, Schonger, & Wickens, 2016). All of these are well-suited, comprehensive, stand-alone experimental software packages that cater to the particular needs of researchers in their respective domain. However, these packages are generally limited in their set of features, and do not offer the extensibility that users of the general-purpose tools enjoy: Researchers are limited in their choice of dependent variables, and forgo recent advances in the graphical design of displays and stimuli, since many dedicated packages for interactive experiments rely on scripting languages or configuration dialogs to construct studies.

These technical limitations on both sides have thus presented a difficulty for researchers aiming to bridge the gap between both fields while maintaining the standards expected in the literature. Based on this divergence in the available tools, we intuit that research in both fields, and particularly at their intersection, would be best served by a modular approach that combines the flexibility of general-purpose experimental software with the features necessary for interaction between participants. With such tools, researchers can combine interactive designs with the wide range of options and extensions available in modern experimental software. To give an example of this modular approach, over the last years an ecosystem has sprung up around several Python-based libraries for stimulus display and response collection, starting with PsychoPy (Peirce, 2007), PyEPL (Geller, Schleifer, Sederberg, Jacobs, & Kahana 2007) and, more recently, Expyriment (Krause & Lindemann, 2014). Building upon these low-level software tools, the OpenSesame graphical experiment builder (Mathôt et al., 2012) offers a powerful and easy-to-use drag-and-drop visual interface for building experiments. As an example of a third-party library within this ecosystem, the PyGaze package (Dalmaijer, Mathôt, & van der Stigchel, 2014) gives users of the aforementioned tools access to eye-tracking, across the various tools for experimental design, and all major vendors of eye-tracking equipment. Using a library like PyGaze, researchers can thus use similar concepts and code not only across different eye-trackers, but also choose the software package they prefer to build their experiments, be it out of familiarity, ease of use, or because their software of choice offers some particular feature.

In summary, there is growing interest in studying strategic interactions not only with regard to choices but also process measures. With this, the need arises for technical options that allow researchers to realize complex interactive

paradigms, and enable participants to communicate and to make joint, interdependent decisions. In this paper, we present an open-source, modular, platform-agnostic approach to interactive experiments. We outline and demonstrate a library that makes such experiments available to the growing ecosystem of Python-based experimental software (and beyond), with both a code-based and a drag-and-drop visual interface (through OpenSesame).

In the following, we illustrate the use of the library with a short tutorial that presents its central features. For ease of exposition, we use the OpenSesame plugin provided with the library. However, the same functions can be accessed through pure Python code in any of the Python-based libraries listed above (examples are provided in the online documentation). Subsequently, we outline psynteract's inner workings, so that interested researchers and developers can extend it to additional experimental software. As we hope to demonstrate, the mechanisms underlying our library are deliberately designed so that plugins for other software packages should be easy to create – all that is needed is the ability to make network requests using HTTP, and decode JSON data, both of which are commonplace in modern programming environments.

Basic functionality

At the most basic level, psynteract provides only four functions, building blocks through which complex interactive experiments can be assembled. First, as a precondition for all following interactions, clients *connect* to a central hub (the server), which gathers and distributes the data of all clients. Second, clients can *push* data regarding their own state to the server, and, third, download, or *get*, the data of all other connected clients. Finally, clients can be programmed to *wait* until a specific precondition is met on part of all other clients, a subset of clients, or the server. For example, a client program can be instructed to wait until the experiment is started on the server, until all other clients signal that the corresponding participants have completed the instructions or a certain number of trials, or until a dyad partner has made her choice. When the library is not performing one of these functions, it is inactive, and requires only minimal computing resources.

A group of participants interacting in parallel, connected by the client software, constitutes a *session*. A session is created and coordinated using a *control panel* operated by the experimenter. This browser-based interface displays the state of the experiment in real time, throughout its entire course. Experimenters can create and start a session, review clients as they connect, monitor the progress of individual participants, and diagnose any problems that might occur. After completion, the data can be archived.

When clients connect, they join the most recent open session, and coordinate their activity with all other clients logged into the same session. As the experiment unfolds, clients send their data (e.g. previous decisions, current state, etc.) to the server, where it can be accessed by all other clients in the same session. Clients are expected to perform computations based on this data independently: If a participant's payoff depends on the choices of her team members over the course of the experiment, the client software takes all these values into account and computes a final sum, rather than being assigned a value from the server.

Psynteract is flexible in accommodating a broad range of designs, thus supporting a wide variety of paradigms. Participants can interact in *groups* of arbitrary size, which can be reassigned at random (a so-called stranger design), or chosen so that any pair of participants interacts only once (a perfect stranger design). Within the groups, participants may also be assigned different *roles* that determine their experience during the experiment. The allocation of participants to groups and roles is computed when the experiment is started, depending on the number of connected clients as well as the desired group size and design. In addition, because it is often challenging to bring together an exact number of participants in the laboratory, it is, if desired, possible to enable a *ghost mode* such that excess participants can 'piggyback' onto others, so that one client may inherit the partners of another and receive the same input, without in turn affecting the interaction of the other players.

A networked setting involving many connected computers increases the risk of technical malfunctions. The design of psynteract provides several safeguards against connection problems, using robust networking protocols for data transmission and establishing a connection to the server only when data is transferred, with clients acting entirely independently between exchanges of information. Accordingly, our experience with the library has shown it to be very stable. However, it is helpful to have a contingency in the event of client malfunctions or permanent loss of the physical connection between single clients and the server. For this case, psynteract offers the option to 'replace' clients during the experiment: Any connected client can be designated a stand-in for a malfunctioning client from the control panel, so that requests for data are transparently re-routed to the stand-in from there on, and the malfunctioning client is subsequently ignored. For many designs, this means that a running session can continue even if one or more clients have to be excluded¹.

¹ Because replacements remain constant for the remainder of the experiment, this mechanism works where no roles are assigned, or where roles remain constant over the course of the experiment.

Usage

To briefly demonstrate these basic concepts, we now show how a basic interactive game might be built from scratch in OpenSesame. As will become clear, this can be achieved almost entirely without code, using only the visual interface provided. The completed example is bundled with the library, as are several variants and code-based equivalents of this and multiple additional paradigms. Extensive documentation regarding the installation of psynteract and the use of its features is available online at <https://github.com/psynteract/psynteract-docs/>.

Our goal is to build an *ultimatum game* (Güth, Schmittberger, & Schwarze, 1982; Güth & Tietz, 1990) which is a well-known economic game for two players, a *proposer* and a *responder*. It is sequential in nature, in that both players make choices in turn. First, the proposer is given an *endowment*, often a monetary value. She may then split this endowment between herself and the responder as she sees fit. The responder is offered his share and may choose to accept it and receive his share, or reject the offer, in which case both players leave empty-handed. It could be argued from the standpoint of classical economics that proposers should keep the entire endowment to themselves, and responders should be happy with any offer no matter its size, but participants share their pies, and unfair splits are largely rejected (see Oosterbeek, Sloof, & van de Kuilen, 2004, for a review).

The psynteract library is available from its repository at <https://github.com/psynteract>, from where it can be freely downloaded. The network capabilities require an instance of Apache CouchDB installed either directly on the computer used to build the experiment, or (for use in a laboratory) accessible over a network connection. For the visual interface used in the following, it depends on OpenSesame (at least version 3.1 at the time of writing), and psynteract's OpenSesame integration, which needs to be installed separately. After installation, a psynteract entry appears in OpenSesame's menu bar, and the main psynteract functions outlined above become available for inclusion in the experiment by drag-and-drop (dark icons in Fig. 1). From the menu, the psynteract backend can be transferred to a CouchDB instance by indicating the CouchDB URL. This provides the server with which all clients later communicate, and need only be done once, as the backend can be reused across experiments. It is also possible to host several distinct psynteract instances on a single CouchDB server, for example to isolate data from different experiments or to run multiple experiments in parallel.

When creating an experiment, users create a sequence of items that run one after another (Fig. 2). The first step is to include a *connect* item at the beginning of the experiment.

After entering the server's URL and the database name, the researcher may select from the designs outlined above, and specify the group size and any roles present within the group (see Fig. 1 for the settings used in our tutorial example). Finally, the number of distinct groupings required will determine how many (re)allocations the system will prepare. When the experiment is run, the experimenter logs into the backend (Fig. 3) and can observe the individual clients connecting to the common session. The clients will pause at the connect item, proceeding in unison when the session is started by the experimenter.

Subsequent to the initial connection, participants will typically be free to peruse instructions, and possibly examples of the task to follow. These can be constructed using the visual tools included in OpenSesame. The generation of displays is documented within OpenSesame, and several tutorials are available, so we will not cover these steps in detail.

Following the instructions, a *wait* item synchronizes participants by pausing until the last participant has completed the introduction and reached this stage. Participants then begin with the actual task simultaneously.

At this point, the interaction of OpenSesame and the psynteract library comes into play. OpenSesame stores all of its data in variables, where it is logged and can be retrieved later, for example to determine the progress of the experiment. Having allocated groups and assigned roles at the beginning of the experiment, our library extends this powerful system by making it possible to share variables between clients, and providing additional variables to each client representing its assigned partners and role. For example, all players will receive a preset `current_role` variable, containing one of the roles specified at the onset. Using this variable, the allocation screen is shown only to the proposer, and the responder is bidden to wait at this stage (see Fig. 2).

The experimental task is, of course, the heart of any study. In the case of the ultimatum game, the proposer selects a split of a (monetary) resource between herself and her partner. For simplicity, our example uses the built-in multiple choice screen to present a range of possible allocations. However, many more possibilities are offered by OpenSesame, ranging from basic keyboard and mouse responses to complex forms. These options can be customized further and extended considerably using Python code.

Using a *push* item, the proposer's choice is transferred to the server and made available to her counterpart. A subsequent *wait* item ensures that choice and transmission are completed before both players *get* the other's collected data. Following retrieval, psynteract makes all data provided by other clients available through OpenSesame variables – for example, the content of the `proposal` variable collected above will be available to the responder as

Fig. 1 The psyinteract items as they appear in OpenSesame after installation (top right of the item toolbar). To the right, the settings for the example experiment described are already set in the connect item

partner01.proposal, and the client can use it in displays, or perform computations based on this variable. In our example, a short and straightforward code is necessary to translate the proposal from the label of the chosen multiple-choice option into the corresponding numeric representations of both players' gains.

The subsequent items in the experiment pertain to the responder's acceptance or rejection of the proposal. This time, it is the proposer who waits until a decision has been made, and through the push-wait-get sequence described above, the data is once more distributed, and the outcome shown to both participants.

This example shows the basic functionality of the psyinteract package, but many extensions and variations are possible: For example, the ultimatum game could be translated into a repeated variant by adding a loop, and the mere addition of a *reassign* item at the end of the loop would shuffle partners and roles after each iteration. Finally, the repeated push-await-get sequences illustrate the utility of the *communicate* item, which combines these steps into one (i.e., both of the push-await-get sequences could be replaced by a single *communicate* item each).

Although we have only scratched the surface of what is possible in OpenSesame, we hope that a few features have become salient: In particular, the experiment was constructed almost entirely using built-in visual tools,

with an absolute minimum of code. We are convinced that many familiar interactive paradigms (games, auctions, etc.) can be built in a similar fashion, quickly and easily. In addition, these paradigms can be enhanced using the plugins already available for OpenSesame – for example the aforementioned PyGaze (Dalmaijer et al., 2014) for eye-tracking and Mousetrap (Kieslich & Henninger, 2016) for mouse-tracking, both of which can be added in a similar drag-and-drop manner. Going beyond the designs already in use, we believe that the mechanisms described are sufficiently general to allow for very complex novel designs. Finally, the addition of code allows for even more fine-grained customization of various aspects of the experiment.

Technical background

On a technical level, communication in psyinteract is based on the ubiquitous HTTP protocol (Fielding et al., 1999), which is known for powering the World Wide Web. Clients exchange data with a central server that stores the information of all clients, and allows any single client to access the information provided by others. Forgoing direct communication between clients drastically simplifies the structure of the library and improves stability, because clients need not

Fig. 2 The entire sequence of items that make up the ultimatum game constructed in the example. The psyinteract components have dark icons. Some items are shown only to players with a certain role, as determined by the condition set in the run if column

respond to requests themselves, and the network connection can lie dormant if it is not needed.

CouchDB, an open-source database that can communicate using the HTTP protocol, provides the server through which all communication takes place. All information is saved within this database: As the experiment progresses, clients push their data to the central database, and may access the data of other connected clients by requesting it from there. The database can also notify clients of changes

made by other clients within the same session, which is the foundation of the wait functionality and the temporal synchronization of clients. Finally, CouchDB hosts the control panel and updates the information shown to the experimenter.

Access to the database is relatively rapid. A complete sequence of pushing data, waiting for an update on part all connected clients, and retrieving a document from the database requires 156ms of roundtrip time on average

Session archive

These are your currently connected clients

Name	ID / Metadata	Data	Options
1	6cf90281aada5feb2c6126963f2d911	<pre>{ "os_status": { "wait_instructions_finished": 1, "wait_proposal": 1, "wait_decision": 1 }, "os_variables": { "decision": "accept" } }</pre>	⚙️
2	6cf90281aada5feb2c6126963f2fee6	<pre>{ "os_status": { "wait_instructions_finished": 1, "wait_proposal": 1, "wait_decision": 1 }, "os_variables": { "proposal": "\$6 for you, \$4 for your partner" } }</pre>	⚙️

Fig. 3 The psynteract experimenter backend, from which the experiments' progress is monitored and can be controlled. The display shows two connected clients at the end of the experiment described. At this point, the only action left for the experimenter is to archive the session (the blue button at the top right, which had previously provided the opportunity to start and close the session). The clients are shown in the table, where they are identified by their human-readable name and the technical identifier specified by the database. To accommodate the diverse types of data that can be stored, a JSON representation

of the client data is provided. Within the JSON data, two data entries are visible. The first indicates the client's status within the experiment: here, all wait points specified in Fig. 2 have been passed once. Second, the two clients make different variables available that reflect their behavior in the experiment: The proposer has suggested a split, and the responder has accepted it. Finally, additional options are available to the experimenter through the button on the right, namely a low-level view of the client document within the database, as well as the option of replacing a malfunctioning client

($SD=18\text{ms}$) for ten connected clients². The last client to update its document incurs an additional delay of 61ms

²These figures are based on benchmarks performed in the University of Landau Cognition Lab, which is equipped with off-the-shelf desktop PCs for clients and server, all connected by cabled network. To measure lag in a worst-case scenario, we simulated all clients on a single computer, and synchronized the database access so that all clients hit the server at precisely the same time. Each latency represents the time from the initiation of the connection to the server up to the completion of the request (or request sequence). Statistics represent 1000 repetitions of each simulation. To simulate variability in response times on part of the clients during the waiting interval, one client was randomly chosen to delay its update for a pre-specified duration before allowing all clients to continue. We subtracted this known delay from the overall

on average. It is important to note, however, that these lookup times in no way affect the performance of the experimental software running on the clients between database transactions. This marks a departure from the currently existing software for interactive experiments, where all displayed information passes through the server and client performance is therefore constrained by the network.

request duration to isolate the latency due to processing and communication. The exact values will depend on the hardware specifications of the clients and the server. Thus, we provide the benchmarking code as part of the psynteract Python package, so that latencies can be measured in each laboratory anew.

```

{
  "_id": "6cf90281aada5febf2c6126963f2fee6",
  "_rev": "9-f40d0a94d94e4150d5261727fe3cf351",
  "timestamp": "2016-05-06T10:38:37.987Z",
  "type": "client",
  "session": "6cf90281aada5febf2c6126963f2c47e",
  "name": 2,
  "group": "default",
  "design": {
    "roles": [
      "proposer",
      "responder"
    ],
    "type": "perfect_stranger",
    "group_size": 2,
    "groupings_needed": 1,
    "ghosts": false,
    "replacements": true
  },
  "data": {
    "os_status": {
      "wait_instructions_finished": 1,
      "wait_proposal": 1,
      "wait_decision": 1
    },
    "os_variables": {
      "proposal": "$6 for you, $4 for your partner"
    }
  }
}

```

Fig. 4 JSON document corresponding to the state of a single client at the end of the experiment described in the previous section. The design section outlines the design expected by the client, and represents the settings shown in Fig. 1. The data section contains all the information that is shared between clients. The nested `os_variables` section contains the OpenSesame variables pertaining to the choices made, and the `os_status` section contains the waiting points that have been passed over the course of the study. These entries are generated automatically by the OpenSesame plugins, but the `data` section is in no way limited to the entries shown, and can be filled with arbitrary information. According to the session document shown in Fig. 5, this client takes the role of the proposer, which is reflected in the variables shared: Its counterpart shares only a decision variable which contains one of values `accept` or `reject` depending on the respondent's decision

All data pertaining to a session is stored as ‘documents’ in the database. Each client, and the session as a whole, is represented by one such document. The session document is accessed (and can be modified) through the browser-based control panel by the experimenter, while each client updates its own document, saving relevant data throughout the experiment.

Data within each document is stored and exchanged in the text-based JSON format, which is capable of holding

```

{
  "_id": "6cf90281aada5febf2c6126963f2c47e",
  "_rev": "3-a7df6fc6756d9f98498fc8be10ad4062",
  "type": "session",
  "status": "closed",
  "opened": "2016-05-06T10:37:31.968Z",
  "replace": {
  },
  "groupings": [
    {
      "6cf90281aada5febf2c6126963f2fee6": [
        "6cf90281aada5febf2c6126963f2d911"
      ],
      "6cf90281aada5febf2c6126963f2d911": [
        "6cf90281aada5febf2c6126963f2fee6"
      ]
    }
  ],
  "roles": [
    {
      "6cf90281aada5febf2c6126963f2fee6": "proposer",
      "6cf90281aada5febf2c6126963f2d911": "responder"
    }
  ]
}

```

Fig. 5 State of a session after completion of the experiment described above. The design parameters set by the client shown in Fig. 4 are reflected in the assignment of participants: There is a single grouping that maps each participant onto the other. This section would grow in size both with an increasing number of participants and additional rounds between which participants are reallocated. Likewise, the `roles` section contains a single mapping of participants onto roles, which would expand similarly to accommodate more participants, and distinct reallocations of roles

almost any data, be it numbers or text, as well as collections of these, such as arrays or hashes. Users of the libraries do not, however, interact with the JSON representation directly, as the data are automatically converted into the data structures of the client's environment: The variables described above if the experiment is built using OpenSesame, and dictionaries if raw Python is used.

To give just two examples, Figs. 4 and 5 show the JSON documents representing the session and an exemplary client after the demonstration experiment constructed above. Each entry in the document consists of a value containing the data of interest, and a text key under which the data can be accessed. Regardless of its type, each document has an `_id` field, which is a random string that uniquely identifies the document. Likewise, the `_rev` field represents the version of the document, which is denoted by a counter and a randomly generated annex. Finally, the `type` field is shared by all documents, and classifies a document as representing either a session or a client.

Sessions and clients diverge regarding the additional properties present in their respective documents. Clients provide a `name` field, which is intended as a human-readable identifier, such as the subject or workstation id. Their `session` field contains the id of the session they are connected to. Most importantly, the `data` field contains all data a client chooses to share, again in a key-value format. In the case of our OpenSesame-based experiment, the selected OpenSesame variables are stored here. In principle, however, the field can hold arbitrary information as long as it can be represented in a text-based format, and users who choose to program their experiments through code can decide freely which data is saved and how. Lastly, clients specify the `design` they expect in the corresponding field. Sessions provide a `status` field, which marks them as either open, running, closed or archived. The `opened` field contains a timestamp of their creation date. Most importantly for our purposes, the `groupings` field assigns the clients into pairs or groups. Here, each client id is mapped onto its partners' ids, and as many such mappings are generated as requested (and allowed by the chosen design). In the given example, because only two clients are connected, only one mapping is possible, and both clients are assigned to their respective counterpart (more involved examples are provided in the online documentation). Likewise, the `roles` field maps clients onto their roles, which can also be reallocated multiple times if required.

For the most part, however, these inner workings are invisible to the end user, and rather handled entirely by the library, which encapsulates these communication flows and provides researchers with the simple interface described and demonstrated above. Because of the platform-independent nature of HTTP, researchers can implement their own library for the platform of their choice: Most if not all major programming languages provide support for communication via HTTP and data serialization via JSON, so we believe that this is a manageable task (at the time of writing, the Python implementation is less than 450 lines of code, including extensive comments and documentation). In return, researchers can make use of the server and its session management interface, and need not create their own backends for supervising clients.

Because the communication flows and data storage formats are standardized, a single experimental session can be comprised of several clients using entirely different software stacks. For example, the majority of participants might be interacting via tablets using the OpenSesame Android extension, while one participant might complete the study at a more traditional workstation, connected to an eye-tracker, other neurophysiological hardware, or using many of the other options available for OpenSesame and other experimental software packages. Likewise, because the HTTP

protocol is the lifeblood of the Web, the mechanisms lend themselves naturally to use in browser-based, online experiments³.

Discussion

The psynteract package enables users of several popular experimental programs across multiple platforms to easily extend their repertoire to include the burgeoning class of interactive studies. Such studies can be built using either the easy-to-use graphical interface provided by OpenSesame, or through any code-based Python library, allowing for the efficient development of experiments using familiar tools. As an open-source package, the library is freely available for use and modification. In addition, the open and flexible protocol can be incorporated into many other platforms and programming languages, using their features and respective strengths to gain more knowledge about strategic interactions.

As noted above, there are several tools available that provide similar features, giving researchers freedom of choice depending on their requirements. We maintain, however, that psynteract is unique in several respects, and goes beyond the tools available in several ways: First, experiments can be assembled using a WYSIWYG graphical interface (via OpenSesame), where previous tools required learning a custom scripting language or (in the case of z-Tree) use of configuration dialogs to customize the study. Coding of experiments is also possible using Python alone, with which many researchers are already familiar. Second, where previous solutions were integrated, complete, and all-encompassing, psynteract is designed to be a component of larger, modular software. This makes available many combinations of tools, enabling new types of data collection (in particular, process measures such as eye- and mouse-tracking), as well as allowing for new flexibility in the design of stimuli.

Because of psynteract's unique design, a direct comparison to the software packages named so far is difficult, as the stimulus display and data collection features for psynteract depend on the experimental software it is combined with. With regard to the design options for interactive experiments, psynteract supports all commonly used participant allocation schemes as well as varying group sizes and roles.

³Researchers willing to go beyond a closed laboratory environment should note that additional security measures are required to ensure confidentiality of participant's data in transit. In particular, all communication between clients and server can and should be encrypted and access-controlled. The online documentation includes pointers on how to strengthen security in such a scenario.

We have successfully used early versions of psyinteract for our own research (e.g. Kieslich & Hilbig, 2014; Hilbig, Thielmann, Klein, & Henninger, 2016), and have tested it extensively.

We are looking forward to fellow researchers using the creative freedom now at their disposal, and exploring the possibilities available with this software. As a final note, we encourage future tool-builders in the social sciences and economics to build on available tools, creating modular software that integrates with existing packages and is easily extended to others. As is visible from Python-based tools for experiments, and R for data analysis, ecosystems of open software provide users with consistent, easy-to-use interfaces, and developers with solid foundations to build upon. We hereby release psyinteract as a public good, hoping that it can provide such a building block for future research. As with the interdependent decisions it is designed to study, we believe that it will benefit from the involvement, suggestions and contributions of researchers with a wide variety of backgrounds, experiences and needs, all of whom and which we warmly invite.

Acknowledgments The authors would like to thank Anja Humbs at the University of Mannheim Chair of Experimental Psychology, Luisa Horsten and Sina Klein at the University of Landau Cognition Lab, and Hosam Alqaderi and Susann Fiedler at the Max Planck Institute for Research on Collective Goods, Bonn, for testing the software and providing valuable feedback. This work was supported by the University of Mannheim's Graduate School of Economic and Social Sciences funded by the German Research Foundation.

References

- Chen, D. L., Schonger, M., & Wickens, C. (2016). oTree—An open-source platform for laboratory, online, and field experiments. *Journal of Behavioral and Experimental Finance*, 9, 88–97. doi:10.1016/j.jbef.2015.12.001
- Dalmaijer, E. S., Mathôt, S., & van der Stigchel, S. (2014). PyGaze: An open source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. *Behavior Research Methods*, 46(4), 913–921. doi:10.3758/s13428-013-0422-2
- Fiedler, S., & Glöckner, A. (2015). Attention and moral behavior. *Current Opinion in Psychology*, 6, 139–144. doi:10.1016/j.copsyc.2015.08.008
- Fiedler, S., Glöckner, A., Nicklisch, A., & Dickert, S. (2013). Social Value Orientation and information search in social dilemmas: An eye-tracking analysis. *Organizational Behavior and Human Decision Processes*, 120(2), 272–284. doi:10.1016/j.obhdp.2012.07.002
- Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext transfer protocol – HTTP/1.1. RFC Editor.
- Fischbacher, U. (2007). z-Tree: Zurich toolbox for ready-made economic experiments. *Experimental Economics*, 10(2), 171–178. doi:10.1007/s10683-006-9159-4
- Geller, A. S., Schleifer, I. K., Sederberg, P. B., Jacobs, J., & Kahana, M. J. (2007). PyEPL: A cross-platform experiment-programming library. *Behavior Research Methods*, 39(4), 950–958. doi:10.3758/BF03192990
- Güth, W., Schmittberger, R., & Schwarze, B. (1982). An experimental analysis of ultimatum bargaining. *Journal of Economic Behavior and Organization*, 3(4), 367–388. doi:10.1016/0167-2681(82)90011-7
- Güth, W., & Tietz, R. (1990). Ultimatum bargaining behavior: A survey and comparison of experimental results. *Journal of Economic Psychology*, 11(3), 417–449. doi:10.1016/0167-4870(90)90021-Z
- Hertwig, R., & Ortmann, A. (2001). Experimental practices in economics: A methodological challenge for psychologists? *Behavioral and Brain Sciences*, 24, 383–403.
- Hilbig, B. E., Thielmann, I., Klein, S. A., & Henninger, F. (2016). The two faces of cooperation: On the unique role of HEXACO agreeableness for forgiveness versus retaliation. *Journal of Research in Personality*, 64, 69–78. doi:10.1016/j.jrp.2016.08.004
- Janssen, M. A., Lee, A., & Waring, T. M. (2014). Experimental platforms for behavioral experiments on social-ecological systems. *Ecology and Society*, 19(4). doi:10.5751/ES-06895-190420
- Kieslich, P. J., & Henninger, F. (2016). Mousetrap: Mouse-tracking plugins for OpenSesame (Version 1.2.1). doi:10.5281/zenodo.163404
- Kieslich, P. J., & Hilbig, B. E. (2014). Cognitive conflict in social dilemmas: An analysis of response dynamics. *Judgment and Decision Making*, 9(6), 510–522.
- Krause, F., & Lindemann, O. (2014). Expyriment: A Python library for cognitive and neuroscientific experiments. *Behavior Research Methods*, 46(2), 416–428. doi:10.3758/s13428-013-0390-6
- Mathôt, S., Schreij, D., & Theeuwes, J. (2012). OpenSesame: An open-source, graphical experiment builder for the social sciences. *Behavior Research Methods*, 44(2), 314–324. doi:10.3758/s13428-011-0168-7
- Oosterbeek, H., Sloof, R., & Van de Kuilen, G. (2004). Cultural differences in ultimatum game experiments: Evidence from a meta-analysis. *Experimental Economics*, 7(2), 171–188. doi:10.1023/B:EXEC.0000026978.14316.74
- Peirce, J. W. (2007). PsychoPy – Psychophysics software in Python. *Journal of Neuroscience Methods*, 162(1–2), 8–13. doi:10.1016/j.jneumeth.2006.11.017
- Psychology Software Tools Inc (2012). E-Prime (Version 2.0). Retrieved from <http://www.pstnet.com>.
- Rand, D. G., Greene, J. D., & Nowak, M. A. (2012). Spontaneous giving and calculated greed. *Nature*, 489(7416), 427–430. doi:10.1038/nature11467
- Seithe, M., Morina, J., & Glöckner, A. (in press). Bonn eXperimental System (BoXS): An open source platform for interactive experiments in psychology and economics. *Behavior Research Methods*. doi:10.3758/s13428-015-0660-6
- Stewart, N., Gächter, S., Noguchi, T., & Mullett, T. L. (2016). Eye movements in strategic choice. *Journal of Behavioral Decision Making*, 29(2–3), 137–156. doi:10.1002/bdm.1901
- Van Lange, P. A. M., Joireman, J., Parks, C. D., & Van Dijk, E. (2013). The psychology of social dilemmas: A review. *Organizational Behavior and Human Decision Processes*, 120(2), 125–141. doi:10.1016/j.obhdp.2012.11.003