Cree: A modern toolbox for readymade economic experiments

Jonas K. Sekamane

Abstract

Cree is a toolbox to design and run economic experiments. *Cree* places very few restrictions on the devices facing subjects, thus *Cree* experiments can be run in a myriad of settings. This opens up a new path for researchers, however the path is not without obstacles. This paper describes how many of these obstacles are alleviated though the technical design of *Cree*. The main remaining obstacle is the selection process, especially if subjects can self-select into experiments.

Supervised by Ulrik Haagen Nielsen. *June 23, 2015*

1. Introduction

This paper introduces a modern toolbox for readymade economic experiments called *Cree*. *Cree* takes advantage of the great advances in technology, in particular the new types of devices (smart-phones, tablets) and the availability of general-purpose software libraries.

The great merit of *Cree* is the very few restrictions it places on the equipment facing subjects. Few restrictions clear the way for much broader participation, strengthening the external validity of any experiment. *Cree* experiments can run in a myriad of settings, including laboratories, but also in *extra-laboratory* settings such as classrooms, workplaces, or over the Internet (Charness et al., 2013). With *Cree* the computer-equipped laboratory is no longer a necessity – lowering the overall costs of conducting economic experiments. The researcher can still choose to provide subjects with devices, but can just as easily let subjects use their own devices.

Cree is build using web technologies. Content is structured using the markup language *HTML*. The layout and presentation across different screen sizes is archived with the style sheet language *CSS*. And all logic is constructed using the programming language *JavaScript*. All web browsers interpret these cornerstone languages and render pages accordingly. Subjects participate in *Cree* experiments through a web browser. Because *Cree* is fundamentally native to the web browser, it avoids many of the restrictions that other toolboxes suffer from.

In addition there is a vibrant ecosystem surrounding these web technologies. *Cree* exploits this ecosystem and takes full advantage of the software libraries that exist. Appendix A lists all the libraries used by *Cree* and gives a short description of each. Building on top of existing technologies provides stability, flexibility, and makes it easy to set up and run economic experiments in *Cree*. Fore instances, *Cree* uses the software library *Node.js* to run the server and handle networking issues. The asynchronous architecture of *Node.js* gives *Cree* the flexibility to handle real-time events, which many other toolboxes are not capable of. With this flexibility *Cree* can run everything, from simple dictator game experiments, to highly sophisticated auction experiments. *Cree* is a framework that provides the tools to handle many of the otherwise mundane, complicated or manually tasks required to set up and run economic experiments.

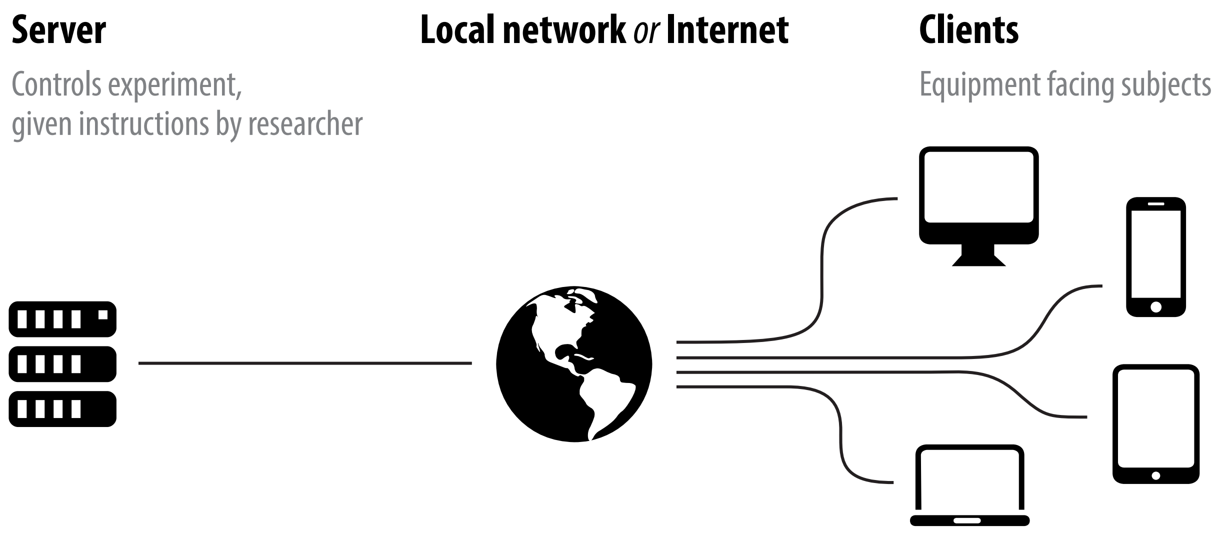
Technological advances opens up a new path, however the path is not without obstacles. This paper explores and discusses how to handle these obstacles. Some obstacles are alleviated through appropriate technical design of the toolbox. Other obstacles requires actions taken by the researcher. While this paper deals with internet experiments in general, it continuously refers to studies that use subjects from Amazon's *Mechanical Turk*. This is not to highlight or promote the use of *Mechanical Turk*, it is simply because the bulk of internet experiments use this system.

2. Requirements

2.1. System requirements

Cree requires a server, ie. any computer with *Node.js* installed. *Cree* has several dependencies – because it is built on general-purpose software libraries – however all of these dependencies are automatically installed, or come pre-bundled, with *Cree*. The server will need to be connected to the devices used by the subject, either through the internet or through a local network (eg. Wi-Fi), see figure 1.

Figure 1: Schematic overview of the network setup with server and clients


Server
Controls experiment, given instructions by researcher
Local network *or* Internet
Clients
Equipment facing subjects

Subjects access *Cree* experiments through a web browser. Any modern web browser will suffice[1], which means that subjects can participate in experiments using smart-phones, tablets, or any computer (Windows, Mac, Linux). Every subject will need a device. The researcher can choose to provide devices, or have subjects use their own personal devices.

2.2. Alternative toolboxes

z-Tree is currently the commonly used toolbox for conducting economic experiments. The development of *z-Tree* started in 1995 and has been updated continuously (Fischbacher, 2007). However the foundation and guiding principles of *z-Tree* is based on the technology that was available two decades ago. Several other toolboxes exist that run economic experiments.

[1]A modern web browser is either: IE version 9+, Chrome current version, Firefox current version, OS X Safari version 5.1+, Opera version 12.1+, iOS Safari version 6.1+, or Android Browser version 4.0+. The browser must allow JavaScript, which all browsers due by default. *Cree* automatically check that these requirements are met, before the subject is allowed to enter the experiment.

Below I divided the toolboxes into four categories based on the restrictions they place on the equipment facing subjects:

Application:	z-Tree, E-Prime, ESI/ICES Software
Requires Python:	UAA-PEET
Requires Java:	BoXS, EconPort, Multistage, Comlab, Aton, JessX, JAuctions, JMarkets, SWIEE
Browser native:	SoPHIE, Willow, jsPsych, Tatool, QRTEngine, WebExp, WEX-TOR, VeconLab, oTree, MWERT

Toolboxes, such as *z-Tree*, are applications. They are written for a specific operating system, and will only run on that specific operating system. All toolboxes in this category require Microsoft Windows. These applications are not easily ported to other computer operating systems. And they will require a complete restructuring – focused on optimising screen sizes and battery life – if they are ever to run on smart-phones or tablets. In addition, these applications need to be downloaded (and installed) on the computer, before subjects can participate. This requirement hinders subjects from using their personal devices.

The second category contains toolboxes written in *Python*. Python is platform independent, so these toolboxes can run on several operating systems, but they require that Python is installed on the equipment facing subjects[2]. These toolboxes also require downloading software.

Toolboxes in the third category are written in *Java*. These are less restrictive, since subjects can participate in experiments though a web browser that has the Java-plugin. Thus there is no need to predownload or preinstall software on the equipment facing subjects. While the Java-plugin is widely distributed, it is not universal[3]. And increasing concerns regarding security issues with the Java-plugin has lead browsers manufactures to implement various warnings and hindrances[4].

The fourth category of toolboxes run natively in the web browser, without the need for any browser plug-ins or predownloaded software. These toolboxes place the same low restrictions on the equipment facing subjects as *Cree*. However, none of these toolboxes, except *MWERT*, currently support simultaneous real-time interaction between subjects – ie.

[2]Python can be packaged into stand-alone applications. These stand-alone applications can run on equipment where Python is not installed.

[3]A Millward Brown survey from 2011 found a penetration rate of 73% on PCs in mature markets. Source: http://www.adobe.com/mena_en/products/flashplatformruntimes/statistics.html. This is also mimicked on the server-side with only 0.1% of websites using Java. Source: http://w3techs.com/technologies/overview/client_side_language/all. The Java-plugin is not supported on iOS or Android. Source: http://www.java.com/en/download/faq/java_mobile.xml.

[4]Eg. due to security concerns by default the Java-plugin is disabled in Google Chrome version 42 (released April 2015). Source: https://www.java.com/en/download/faq/chrome.xml.

it is impossible to design experiments where subjects continuously provide input, while they continuously and simultaneously receives feedback from the other subjects. Some toolboxes support sequential decision making, while others completely lack interaction among subjects. The toolbox *MWERT* is build specifically to run real-time multiplayer experiments on the internet (Hawkins, 2014). It relies on many of the same basic technologies as *Cree*, however *MWERT* takes less advantages of the availability of general-purpose software libraries, making it more cumbersome to develop experiments. Fore instance the "build for scratch" approach manifests itself when the researcher wants to run experiments where more than two subjects interact simultaneously. In *MWERT* the researcher has to rewrite the core of the toolbox, whereas the number of subjects is simply a parameter to be specified in *Cree* experiments. Another limitation is the structure of the stages. *MWERT* has just one stage (disregarding the instructions and the exit-survey). Subjects repeat this single stage a given number of times, whereafter the session ends. *Cree* on the other hand gives researchers the flexibility to build a list of all the stages. These stages can be identical, but need not be. And in between each stage, the researcher can add a synchronisation step such that subjects wait for other subjects to catch up, or they can add a randomisation step such that subjects are shuffled among groups, roles or treatments.

3. Running experiments

3.1. Recruitment

The method used to recruit subjects will depend on the type of experiment and the desired representativeness of subjects. It is commonly assumed that subjects are naïve, eg. encounter the experimental material for the first time, and have no a priori knowledge of the experiment, the purpose, nor knowledge of related experiments. This is not true for experiments conducted over the internet, and is often misapprehended by researchers, as shown by Chandler et al. (2014). They investigate experiments where subjects are recruited through Amazon's *Mechanical Turk* system. *Mechanical Turk* is a crowdsourcing platform of 500,000 workers that perform mostly small tasks and are paid a piece rate. Every worker has an account, and *Mechanical Turk* prevents an account from performing the same task twice. Chandler et al. (2014) find that:

1. subjects have experience from related experiments. By pooling 16,408 observations (or completed tasks) across 132 studies, they find just 7,498 unique subjects – averaging 2.24 observation per subject. The 10% most prolific subjects account for 41% of all observations. In addition Chandler et al. (2014) survey 300 workers on *Mechanical Turk* and find that 56% have previously been exposed to the *Prisoner's dilemma* and 52% to the *Ultimatum game*.

2. subjects share information about experiments. About one fourth of workers in the Chandler et al. (2014) survey respectively report that they know other workers in person, and that they read or discuss *Mechanical Turk* online on blogs and forums. The focus of the conversation is typically not the specific contents of the experiment, but rather pay rates and reputation. Chandler et al. (2014) conclude that researchers should be particularly aware of cross talk, when information can increase financial rewards – such as when using techniques to increase attention (*instructional manipulation checks*, or *got-cha* questions), or when using techniques to deny payment. They suggest that researchers ask subjects how they found the experiment, and monitor discussion boards that refer a lot of subjects to the experiment.
3. few subjects attempt to participate repeatedly in the same experiment. Amazon forbids multiply or *sock-puppet* accounts, and actively enforces this policy. Chandler et al. (2014) find 1.2% of subjects share IP addresses as well as demographic characteristics – eg. possible *sock-puppet* accounts. They attribute the low percentage to the active enforcement by Amazon.

Internet experiments in general and *extra-laboratory* experiments, are likely to face similar issues. Especially considering Goodman et al. (2013) find that, in all the studies they looked at, *Mechanical Turk* subjects are not significantly different from traditional samples, except one study (on risk aversion).

Cree does not include a prescreening system to manage the pool of subjects, so the researcher handles recruitment manually. The selection process is crucial, and researchers should carefully consider the degree to which subjects can self-select into experiments, since this may invalidate the underlying assumption of naïve subjects. *Cree* automatically logs referrals from other websites. And researchers can restrict *Cree* so it only accepts one subject per IP address, creating additional protection against subjects participating multiple times in the same experimental session.

3.2. Coordinated joining

When the researcher initiates the experiment, *Cree* returns the URL addresses that subjects use to join the experiment. Two URLs are displayed, since subjects can participate over the internet and through the local network. URLs should be distributed accordingly. The researcher distributes the URL to participants (via printed hand-outs, e-mail, third-party recruitment system, etc). Subjects join the experiment simply by clicking a link, or by typing the URL into their web browser. *Cree* automatically tests if the server can connect to the internet, if this fails then *Cree* only displays the local network URL[5].

Besides recruitment and distribution of addresses, there is coordination. In experiments where subjects interact with each other in real-time, the researcher must coordinate subjects so they start the experiment simultaneously. The researcher can specify that *Cree* should wait for sufficiently many subjects to join, before starting the session. Ie. the first subject joining will be put on hold, until the last required subject joins. Subjects have limited patience, so the researcher needs to coordinate subjects on a very narrow interval of only a few minutes. There are a couple of different approaches that the researcher can take here. In laboratory experiments it is common to invite more participants than needed for a given session (due to "no shows"). This option is also possible in *Cree*, since *Cree* turns down subjects once a session is filled. There is a semi-continuous option that expands the interval. For instance in the *Prisoner's dilemma* it is common with only two subjects interacting with each other. So instead of waiting for all subjects to join the session, the semi-continuous option will start whenever two new subjects join. However with this option the researcher loses the ability to randomise subjects, ie. pair a subject with another. The lack of randomisation may lead to reciprocity among subjects. There is a third fully-continuous option, where the server simply restarts the experiment after each session. This option can be used in conjunction with fewer subjects per session. If subjects show up late for the first session, they wait for the session to end, hereafter they can join the second session, and so on. All of this continues until the researcher manually stops the experiment.

3.3. Stability

A main priority for researchers is that the software they use to run experiments is stable. In addition to being costly and time consuming, crashes are also detrimental to the researcher's reputation among the participating subjects.

There are two stability objectives; The first is eliminating software bugs, ie. faults causing the software to perform in an unintended or unanticipated manner, and due to the error of the programmer. The second is sufficient error handling, ie. implementing fallbacks to deal with mishaps, so crashes are mitigated, or at the very least an emergency landing is performed.

Both stability aspects require thorough testing and running countless experiments using the software. *Cree* does not have a 20-year track record like *z-Tree*, but nor is it build from scratch. The software libraries and dependencies on which *Cree* is build, have been tested intensively by their respective communities. Some libraries are used daily across millions of websites[6]. This speaks volumes to the infrequency of software bugs in the foundation of *Cree*.

[5] *Cree* does not test if access to the server is blocked by a firewall, nor if the two required ports are open.

[6] *jQuery* is used on 64% of monitored websites. There is no statistics on the *Bootstrap* CSS library, but its

With numerous interconnected and simultaneously used devices, eventually a device will malfunction or the network will suffer a temporary disturbance. *Cree* handles these types of errors by allowing subjects that briefly disconnects, to reconnect and continue the experiment. The researcher can specify how long connections are allowed to be absent, before a subject gets kick from the experiment – the default is 10 seconds. Disconnects and reconnects are logged, so the researcher can later determine if it significantly impacted the results of the experiment. I do not recommend reconnecting subjects after lengthy disconnects, since this would hold up the entire experiment and adversely impact the waiting time of the other participating subjects. In these instances the experiment proceeds without the disconnected subject. When a subject definitively disconnects from a *Cree* experiment, then the other subjects in the same group are redirected to a fallback page. This page informs the subjects about the fall out and explains why they are not able to continue the experiment being one man short. The remaining groups are unaffected by the disconnect, and *Cree* is still able to synchronise and randomise among the remaining subjects and groups. The researcher will have to manually calculate payments for subjects in the group that suffered a untimely disconnect, but *Cree* is still able to calculate payments for subjects in the remaining groups.

If subjects attempt to abort the experiment untimely, *Cree* displays a warning message. The occurrences of these warning messages are logged. In addition *Cree* logs when the subject's web browser is out of focus[7] – this is a proxy measure for inattention. Figure 2 plots the session of one subject. The figure includes connection, disconnection, the stages of the experiment, and periods of inattention (areas shaded red). The full lines mark the beginning of a new stage. The dotted lines are where the subject waits for the other participant.

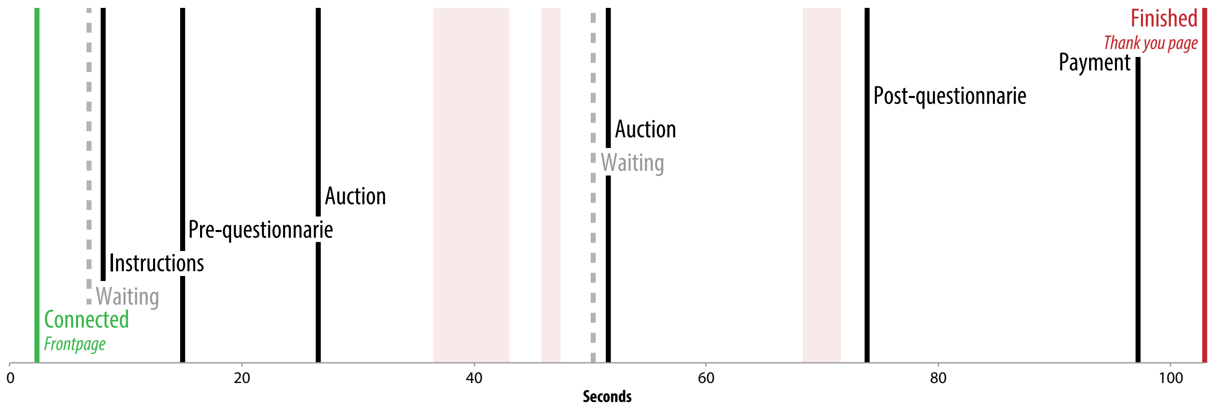
3.4. Duration and supervision

Laboratory experiment can often last an hour or more. There is very little literature on how unsupervised subjects perform in experiments with long sessions. Using *instructional manipulation checks* Oppenheimer et al. (2009) show that student subjects pay less attention when unsupervised. Goodman et al. (2013) argue that *Mechanical Turk* may not be appropriate for long or complicated experiments. They compare the results of their experiment – which lasted about 16 minutes – to that of Paolacci et al. (2010) that only lasted around 5 minutes. Unlike Paolacci et al. (2010) they find that subjects from *Mechanical Turk* are less

complementary JavaScript library is used on 8% of websites. Source: `http://w3techs.com/technologies/overview/javascript_library/all`. While *Node.js* is used as web-server on 0.1% of monitored websites. Source: `http://w3techs.com/technologies/overview/web_server/all`.

[7]ie. when the subject uses a different tab in the web browser, when the browser is minimised or fully obscured by another application, or when the device enters its power saving mode (lock screen on smartphones). Source: `http://www.w3.org/TR/page-visibility/`

Figure 2: Example of a session timeline of a subject



attentive. They test attentiveness using *instructional manipulation checks* at the end of the survey. Farrell et al. (2014) actually run an hour-long experiment on *Mechanical Turk* where subjects fulfil orders (sandwich orders or unspecified product orders). They find that *Mechanical Turk* subjects exert the same amount of effort (or more), and make the same amount of mistakes (or fewer), than student participants. Even in the treatment where *Mechanical Turk* subjects are paid one-fifth of the student participants, where tasks are less intrinsically interesting, and where their wage rate is independent of performance (flat rate of $5). The results of Farrell et al. (2014) may be due to the selection process, ie. the *Mechanical Turk* subjects that volunteer for their experiment, are more motivated by financial incentives, than other individuals. An indication of this is that 73% report ‘extra income’ and 28% report ‘entertainment’ as reasons for using *Mechanical Turk*. Whereas in the Paolacci et al. (2010) study it is respectively 61% and 41%. Similarly in a short survey by Williamson (2014) on *Mechanical Turk* only 28.9% respond that they would be willing to participate in a hour-long follow-up interview for an additional $15.

In sum the researcher needs to carefully consider the selection process, to ensure that subjects are representative and that results have general validity. When subject’s ability to self-select into experiments is restricted, it remains unclear if subjects will participate in hour-long experiments, and it is unclear how attentive they are during these long unsupervised sessions.

There are two alternative approaches, regarding time and duration that the researchers might consider when designing the experiment. One is the approach of Pettit et al. (2014) which speeds up the adjustment process by allowing subjects to continuously change and adapt strategies. They refer to one study where distinctive, long-term implications are observable in rounds lasting only 120 seconds. The other approach, suggested by Charness et al. (2013), is to create long time-series data, ie. by recalling subjects every day, week or month. In a traditional laboratory experiment, this would be time consuming and highly inconvenient for subjects, but less so in extra-laboratory settings such as classrooms or over the internet. Charness et al. (2013, p. 96) argue that *"the classic 10–20 period repetition of the public-goods game may not be a good analog for the expression of preference for tax policy that citizens make when they vote every couple of years"*. *Cree* can execute both alternatives, since it allows real-time interaction among subjects and works in *extra-laboratory* settings.

3.5. Payment

For in-person experiments the researcher can pay subjects in cash. For experiments conducted over the internet payment will require a third-party payment provider. This could be national provides, such as the Danish *MobilePay* or *Swipp*. Or it could be international providers such as *PayPal*. When a third-party payment provider is used, subjects will enter their mobile phone number, email address, or receive a voucher code they can use to claim their payment. At the end of every session *Cree* generates a list with each subject's payment, preferred method of payment, and transfer details.

3.6. Questions

Subjects may have questions regarding instructions or procedures during the experiment. In supervised in-person experiments, the experimenter can walk around and answer these questions. For experiments conducted over the internet, *Cree* can be extend to include a real-time chat function. Using this chat function subjects can privately ask the experimenter questions and receive answers. However this should be a fallback option reserved for special cases, as it may lead to unintended *experimenter effects*, ie. subjects trying to please the experimenter, rather than behaving intrinsically.

3.7. Cheating

Discouraging cheating in unsupervised experiments is difficult. And additional case is needed since subjects participate in experiments using a web browser. First, Goodman et al. (2013) have shown that *Mechanical Turk* subjects search for the correct answers to factual questions on the internet. Subjects look up answers, even when payment is independent of answering correctly. And when subjects are paid for the correct answer, they are more likely to look up the answer. Asking subjects not to use any externals sources reduces the number of subjects that look up the answer, but it does not completely eliminate cheating. Secondly, toolboxes that are native to the web browser are at a disadvantage compared to the toolboxes in the three other categories (discussed in section 2.2), since pages are rendered in the subject's web browser. Thus the source code to the unrendered pages is easily viewable and relatively easy to manipulate (Hawkes, 2011, 25m. 12s.). In *Cree* Subjects cannot cheat by manipulating how the page is rendered, since the server has final authority and controls the experiment. But subjects may attempt to cheat by sending false or manipulated inputs back to the server. It is therefore crucial, when the researcher designs the experiment, that all inputs are validated on the server. In addition since the source code of the page is accessible, it should not contain information that disclose the purpose of the experiment, the treatment, nor optimal strategies – eg. instead of using variables names such as `Option_Nash_Equilibrium` and `Option_Sub_Optimal`, the researcher should use generic names such as `OptionX` and `OptionY`. A third issue is coercion among subjects. Coercion may arise in unsupervised in-person experiments where subjects are seated closely (ie. classrooms, work places, etc.), or in internet experiments where subjects are able to self-select into the experiment, and therefore invite others to join with the aim of manipulating the experiment. In *Cree* the researcher can randomise subjects across groups, roles and treatments, but other than this *Cree* cannot alleviate coercion. If possible it is recommended that in-person experiments be supervised. And researchers should limit the subject's ability to self-select into the experiment, when conducting experiments over the internet.

4. Results

4.1. Expectations

Charness et al. (2013) look across different studies and compare the results from laboratory settings to those from *extra-laboratory* settings. In the *Ultimatum game*, studies in the laboratory have not been able to find a relationship between the stakes (ie. the amount of money that the subjects can split) and the frequency of rejections. Whereas an *extra-laboratory* study in villages in rural India find that higher stakes leads to less frequent rejections. Laboratory studies on the *Public-goods* game yields the same results as a *extra-laboratory* study among card traders at a sports-card event. In the *Trust game*, a study using respectively students in a laboratory and users from the online game *Second Life*, find that subjects from the gaming community are more willing to pay a social-distance premium, and that second movers pass higher amounts back. Another study looks at the closely related *lost-wallet* game. This study finds few differences in the results across their three environments; laboratory, classroom and online. With regard to risk and time preferences, a laboratory study among US students elicit discount rates that are twice as high as credit card rates. Whereas an *extra-laboratory* study using a representative sample of Danish adults elicit discount rates that are above mortgage and auto rates, but below credit card rates. In laboratory settings subjects show high aversion to risk. However a *extra-laboratory* study among evacuees of Hurricane Katrina living in shelters, establish risk-seeking behaviour. Repeating the study a year later with a similar population shows that this behaviour diminishes over time. As noted earlier Goodman et al. (2013) find that *Mechanical Turk* subjects are not significantly different in their behaviour from traditional samples, except in one study on risk aversion.

This short walkthrough indicates that the behaviour observed in laboratories for most part also holds outside of the laboratory. Perhaps the results do not hold in other cultures, or with extremely high payouts (such as rural India). Similarly social preferences differ when subjects self-identify with a community (the gamers). And the researcher should be particularly observant of the selection process when conducting experiments on risk or time preferences, since both age and circumstances will affect experimental results.

4.2. Devices

There is yet no research on how or if results are influenced by the devices used by subjects in experiments. *Cree* records the type of device used by subjects, as well as the screen size. The researcher can use this to post hoc control for differences in the results. Alternatively the researcher can a priori exclude specific devices from participating in the experiment. This may be particular useful in experiments where the subject's reaction time is critical (such as experiments on auctions). I expect that subjects using smart-phones react slower, than subjects on other types of devices. In part due to lower processing power and screen size, but also due to the different input method. The screen, keyboard and mouse are separated on a computer, while on a smart-phone the mouse and keyboard are built into a screen that only covers about one-fourteenth of the area of a computer screen. Subjects with a smart-phone would be at a disadvantage when entering text or numbers repetitively and at length, or when constantly having to switch between keyboard and pointing. Another alternative is levering the playing field, by using similar input method across all devices. Choi et al. (2007) proposed using a graphical interface. For instance, if the subject is asked to allocate an amount to consumption out of a budget of $100. Then instead of entering this amount into a form, there is a line graphically representing the $100 budget, and the subject simply clicks accordingly on the line to input his or her allocation. Similar method has also been implemented by Pettit et al. (2014) in the *ConG* toolbox. *Cree* leverages the JavaScript library *D3.js* to build graphics.

4.3. Control

The researcher can control for differences among subjects post hoc. Chandler et al. (2014, p. 121) are strongly critical of this approach. They estimate that more than a third of all peer-review papers using *Mechanical Turk* drop subjects post hoc for one reason or another. They describe this an *"arbitrary data-cleaning strategy"* and that researchers are *"too eager to exclude workers post hoc"*. They propose a less subjective approach where researchers adjust their selection process, so subjects are excluded a priori. In particular they propose using prescreening surveys that elicit demographic characteristics, psychological attributes, attentiveness, and ability to solve tasks that are crucial to completing the actual experiment. *Cree* does not include a prescreening system to manage the pool of subjects. It offers researchers only a few basic options to exclude subjects a priori. The researcher can choose to allow only one subject per IP address, can provide a list of IP addresses to exclude, and can provide a list of devices to exclude. The researcher will need to employ a third-party prescreening system, for more elaborate a priori exclusion criteria.

5. Designing experiments

5.1. Single versions

The subjects participate in a Cree experiment through a web browser. A Cree experiment consists of HTML pages, that the browsers renders and displays. It is not necessary to build separate versions for respectively smart-phones, tablets and computers. There is only one version and the web browser scales every page so it matches the respective screen sizes.

5.2. Ease

There are some misconceptions that I would like to address.

1. The main purpose of a toolbox is not to offer comfort or convenience to researchers, but to offer the necessary tools. Convenience is secondary.
2. There is a trade-off between usability and flexibility. The programmer bases every improvement in usability on a judgement call. Usability is improved when the set of possible choices are reduced, this in turn reduces flexibility. Thus a toolbox should be evaluated on its ability to balance the trade-off between usability and flexibility.
3. All toolboxes essentially requires some programming skills. It is true that some toolboxes allow researchers to run preprogrammed experiments, without any additional programming. But once the researcher starts building own experiments, or significantly modifies existing experiments, programming is unavoidable. A programming language is nothing more than human logic and instructions in a machine interpretable format, and this is why programming is unavoidable when researchers build logic into their experiments.

The guiding principle behind *Cree* is to offer a solid and highly flexible foundation that gives researchers the necessary tools to carry out their research, irrespective of location and available devices. The current version of *Cree* does require some programming skills. An interface – providing convenience when running preprogrammed experiments and when implementing minor changes – could be added on top of this foundation in a future version. The interface would be a supplement. But with the foundation in order, this would not be at the expense of the general flexibility of *Cree*. *Cree* relies on JavaScript to build logic into experiments. Because JavaScript is a general-purpose programming language and widely used, many of the procedures necessary to build more complex experiments are already easily available. Thereby avoiding the trap noted by Fischbacher (2007, p. 177) when discussing the current limitations of *z-tree*; *"there are situations in which programming is unnecessarily complicated, since z-Tree neither knows procedures nor complex data types ... One of the planned future steps in the development of z-Tree is thus an extension of the programming language to make programming of more complex experiments easier"*. *Cree* uses the CSS framework *Bootstrap*, which insures that pages scale seamlessly across small and big screens. In addition it ensures that pages look identical across different browsers and devices, for given screen sizes. Since Bootstrap handles all this, knowledge of CSS is not need to build elegant experiments that have a consistent look across devices. Some knowledge of HTML is helpful, but not required since the researcher can use third-party drag and drop tools such as *LayoutIt!* or *Jetstrap* to build pages that work with *Bootstrap*[8]. Appendix C gives a broad outline how a Dictator game experiment would look in *Cree*.

6. Conclusion

This paper introduces *Cree*, a modern toolbox to design and conduct economic experiments. *Cree* places very few restrictions on the devices facing subjects, thus relinquishing economic experiments from the grips of the computer-equipped laboratory. In turn economic experiments designed in *Cree* can also run in *extra-laboratory* settings such as classrooms, workplaces and over the internet. While *Cree* provides researchers with new tools it also opens up for new issues and obstacles. This paper shows how many of these obstacles are solved in the technical design of *Cree*, while a few obstacles remain that require action by the researcher. The main remaining issue facing researchers when transferring from a laboratory setting to an *extra-laboratory* setting is not subject attention, subject compensation, nor the quality of the results. It is recruitment and specifically the selection process. If subjects self-select into the experiment, it may change the mindset, behaviour, validity and ultimately the results. A third-party prescreening system is required to manage the pool of subjects. And running prescreen surveys to appropriately exclude subjects on a priori grounds might also.

[8]Source: https://jetstrap.com. Source: http://www.layoutit.com

Appendix A. *Cree* dependencies

There is a vibrant ecosystem and many general-purpose software libraries surrounding the three web technologies; HTML, CSS and JavaScript. *Cree* exploits this ecosystem and takes advantage of these libraries. This increases the stability of *Cree*, provides flexibility, and makes it easy to conduct economic experiments. This section briefly describes each of the software libraries, which *Cree* depends upon.

Node.js: *Cree* uses the software library *Node.js* to run the server and control the experiment. *Node.js* is build on the same JavaScript engine as the Chrome browser, but *Node.js* runs JavaScript on the server. This means that *Cree* uses the programming language JavaScript to induce logic, both at the server level, and at the web browser level. *Node.js* has a asynchronous architecture – so instead of executing the code from the top, one line after the other, *Node.js* binds all code to events and the code is only executed once an event happen (eg. when subjects enter the experiment, or when subjects sends messages, etc.). This asynchronous architecture gives *Cree* the flexibility to effortlessly handle real-time events. *Node.js* is easily installed from http://nodejs.org, and comes with the package manager *npm*.

cloak: This *Node.js* module or package aims to provide a network layer to HTML games. *Cree* uses *cloak* to handle all networking issues, e.g. connecting subjects, queuing subjects, sending messages back and forth between subjects, etc. See http://incompl.github.io/cloak/

knuth-shuffle: *Cree* uses this *Node.js* module to randomise subjects across groups, roles and treatments respectively. The underlying algorithm is known as the *Fisher–Yates shuffle* or the *Knuth shuffle*. And the algorithm is unbiased. See https://github.com/coolaj86/knuth-shuffle

connect: This is the *Node.js* module that *Cree* uses to serve web pages to subjects. See https://github.com/senchalabs/connect

ua-parser-js: *Cree* uses this *Node.js* module to identify and block specific devices and web browsers. See http://faisalman.github.io/ua-parser-js/

underscore: This is a *Node.js* module that extends JavaScript and provides additional useful functions. See http://underscorejs.org

ipify.org: *Cree* calls this *application programming interface* (API), and it returns the public URL address, ie. the URL address that subjects use when joining the experiment over the internet. See https://www.ipify.org.

forever: This command line tool gives *Cree* its fully-continuous option described in section 3.2. If experiments are started using *forever* then the server simply restarts the experiment after each session, and the experiment continues until the researcher manually stops it. See `https://github.com/foreverjs/forever`

Bootstrap: *Cree* uses the CSS framework *Bootstrap*, which insures that pages scale seamlessly across small and big screens. In addition it ensures that pages look identical across different web browsers and devices, for given screen sizes. See `http://getbootstrap.com`

jQuery: This library extends JavaScript and provides additional useful functions to change and manipulate the web pages that subjects see. More specifically it is used to load new pages and collect inputs from subjects. See `https://jquery.com`

D3.js: *Cree* leverages the JavaScript library *D3.js* to build graphics and figures. *D3.js* is used for both the static graphics and the graphical interfaces that subjects interact with and provide inputs through. See `http://d3js.org`

Appendix A.1. Installation

Cree is installed on a computer with Node.js using its package manager (*npm*). To install Cree open *Command Prompt* on Windows, or *Terminal* on Mac/Linux, enter `npm install Cree` and press enter. All of the dependencies are automatically installed, or come pre-bundled, with *Cree*.

Appendix B. Source code

The source code for *Cree* is licensed under the MIT license and available at `https://github.com/jsekamane/Cree`. An example experiment of an English auction is available at `https://github.com/jsekamane/Cree/tree/master/examples/english-auction`. And an example of a Dictator game is available at `https://github.com/jsekamane/Cree/tree/master/examples/dictator-game`. Fieldnotes for this paper are available at `https://github.com/jsekamane/Cree-Paper`.

Appendix C. Dictator game

A *Cree* experiment has three main building blocks – in this example the three are `dictator1.json`, `dictator1.js` and `client/experiment.js`. The file `dictator1.json` specifies the structure of the experiment, ie. which page to show and when. *Cree* requires some basic `utilities`, eg. a page to display when the experiment is full, or a fallback page in case of untimely disconnects. In addition the `dictator1.json` file lists the stages chronologically. In this example most pages are static, but a few pages have a specific `type`, that tells *Cree* to synchronise subjects or randomise them. In this example, the first listed user is always the dictator, but the researcher randomise the list of users, and thus randomises who is dictator. This is done in between the two dictator-rounds (`r1`, `r2`), and the `method` specifies that subjects should not be match with new subjects, but that subjects should shuffle only within their current groups. Static pages may have a `script` that is executed once a user reaches the stage, or once all users reach the stage – this is again determined by the `method`.

Listing 1: The structure of the experiment – `dictator1.json`

```
{
   "utilities": {
      "full": "dictator1/00_full.html",
      "lobby": "dictator1/00_wait.html",
      "decline": "dictator1/00_decline.html",
      "fallback": "dictator1/00_fallback.html"
   },
   "stages": [
      { "name": "instructions", "url": "dictator1/01_instructions.html" },
      { "type": "sync", "name": "wait", "url": "dictator1/00_wait.html" },
      { "name": "r1", "url": "dictator1/02_dictator.html", "script": "dictator",
          "method": "withinRooms" },
      { "name": "wait", "type": "sync", "url": "dictator1/00_wait.html" },
      { "type": "randomise", "name": "Randomise", "method": "withinRooms" },
      { "name": "r2", "url": "dictator1/02_dictator.html", "script": "dictator",
          "method": "withinRooms" },
      { "name": "pay", "url": "dictator1/03_payment.html", "script": "payment",
          "method": "perUser" },
      { "name": "thanks", "url": "dictator1/04_thanks.html" }
   ]
}
```

The file `dictator1.js` is the server script that controls the experiment. The file contains several parameters that configurates the experiment, such as the boolean `autoJoinLobby` and integer `membersPerRoom`. In addition it contains the scripts called from the `dictator1.json` file. Below is a snippet of the `dictator`-script which is called in the beginning of each of the two rounds. It loops through all subjects in the specific group and sends the `choose`-split message to the first subject on the list, while the other subject gets a `wait`-for-split message.

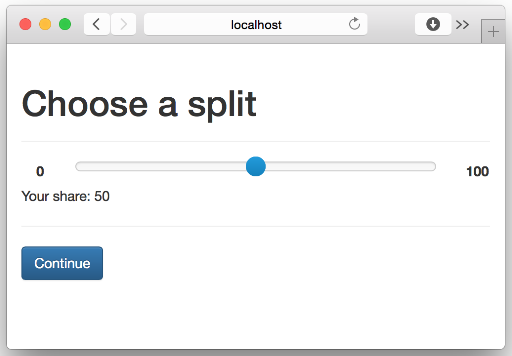
Listing 2: Snippet from the server script – `dictator1.js`

```
global.dictator = function(stage, room){
  // There are only two subjects. The first subject is always the dictator.
  for(var key in room.getMembers()) {
    var user = cree.cloak.getUser(room.getMembers()[key].id);
    if(key == 0) user.message("choose"); // Tell user to chose a split
    else user.message("wait"); // Tell user to wait for split
  }
}
```

The last file `client/experiment.js` is the script that runs in the subject's web browser. It receives messages from the server and changes the subject's page accordantly. Fore instances displaying a slider to the "dictator" from which he or her can choose a split, see figure C.3. Once the subject submits his or her choice, then the `client/experiment.js` file sends a `split` message return to the server, with the amount chosen by the subject, see snippet of this in listing 3.

Figure C.3: The screens of respectively the "dictator" and the other subject

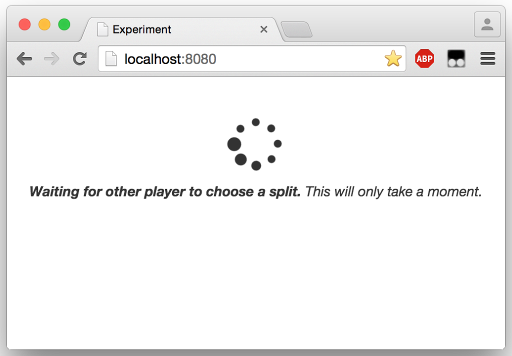
(a) The screen of the "dictator".



Choose a split

Your share: 50

(b) The screen of the other subject.



Waiting for other player to choose a split. This will only take a moment.

Listing 3: Snippet from the client script – `client/experiment.js`

```
$(document).ready(function(event) {
  $('#container').on('submit', '#input-form', function(){
    var amount = Number($('#slider input').val());
    cloak.message('split', amount);
    return false;
  });
});
```

This is how the three main building blocks of a *Cree* experiment work in tandem. `dictator1.json` contains the structure and road-map of the experiment. *Cree* reads this road-map and the configurations in `dictator1.js` and sends messages to the client. The `client/experiment.js` file catches these messages, adjusts pages, waits for input from subjects, and then sends a return message. As seen the researcher can focus on the particulars of the specific experiment, since *Cree* is a framework that provides the tools to handle many of the mundane, complicated or manually tasks required to run economic experiments.

References

Chandler, J., Mueller, P., and Paolacci, G. (2014). Nonnaïveté among amazon mechanical turk workers: Consequences and solutions for behavioral researchers. *Behavior research methods*, 46(1):112–130.

Charness, G., Gneezy, U., and Kuhn, M. A. (2013). Experimental methods: Extra-laboratory experiments-extending the reach of experimental economics. *Journal of Economic Behavior & Organization*, 91:93–100.

Choi, S., Fisman, R., Gale, D. M., and Kariv, S. (2007). Revealing preferences graphically: an old method gets a new tool kit. *The American economic review*, pages 153–158.

Farrell, A. M., Grenier, J. H., and Leiby, J. (2014). Scoundrels or stars? theory and evidence on the quality of workers in online labor markets. *Theory and Evidence on the Quality of Workers in Online Labor Markets (October 4, 2014)*.

Fischbacher, U. (2007). z-tree: Zurich toolbox for ready-made economic experiments. *Experimental economics*, 10(2):171–178.

Goodman, J. K., Cryder, C. E., and Cheema, A. (2013). Data collection in a flat world: The strengths and weaknesses of mechanical turk samples. *Journal of Behavioral Decision Making*, 26(3):213–224.

Guala, F. (2005). Inside the laboratory. In *The Methodology of Experimental Economics*, pages 13–38. Cambridge University Press. Cambridge Books Online.

Hawkes, R. (2011). Rawkets: An inside look at a multiplayer HTML5 game. Google Tech Talk. URL: https://youtu.be/zj1qTrpuXJ8?t=25m12s.

Hawkins, R. X. (2014). Conducting real-time multiplayer experiments on the web. *Behavior research methods*, pages 1–11.

Oppenheimer, D. M., Meyvis, T., and Davidenko, N. (2009). Instructional manipulation checks: Detecting satisficing to increase statistical power. *Journal of Experimental Social Psychology*, 45(4):867–872.

Paolacci, G., Chandler, J., and Ipeirotis, P. G. (2010). Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419.

Pettit, J., Friedman, D., Kephart, C., and Oprea, R. (2014). Software for continuous game experiments. *Experimental Economics*, 17(4):631–648.

Williamson, V. (2014). On the ethics of crowd-sourced research.