

# Building Your Own Experiment

From Design to Data

Limperg Course on Experimental Accounting Research

**Guest Lecture**

June 15, 2020

Christian Peters

TILBURG  UNIVERSITY

# Introduction

Once you have a research question and a design you are ready to build your own experiment.

However, building your own experiment is not an easy task. Where do you start?

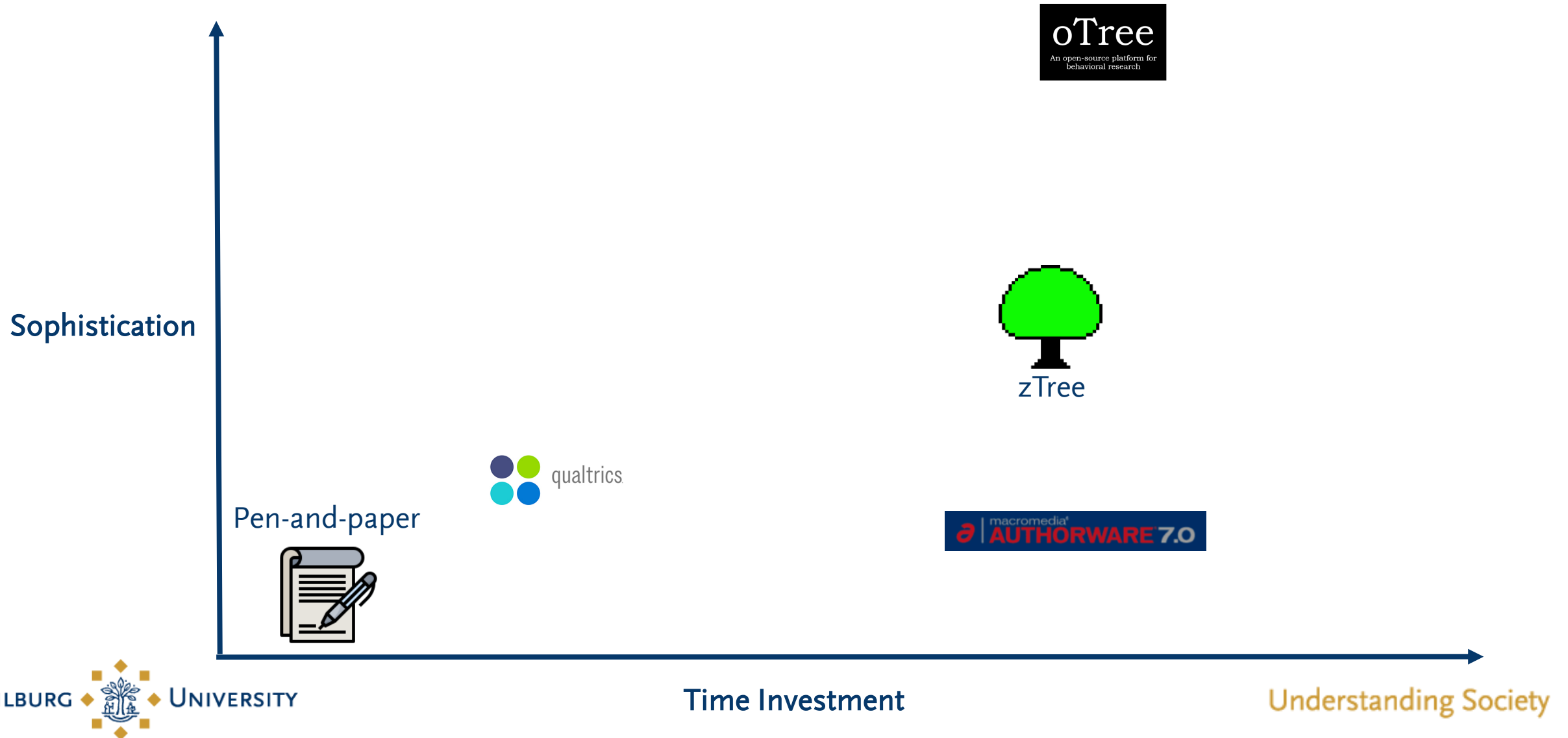
The goal of this lecture is to:

- Give a gentle introduction into programming experiments
- Show you what the different options and trade-offs are
- Show you the building blocks of oTree
- Show you how to deploy your experiment to a server
- Show you how to make your experiment available to online participant pools

This lecture has a companion respository with information on all the details:

<https://github.com/CPHPeters/>

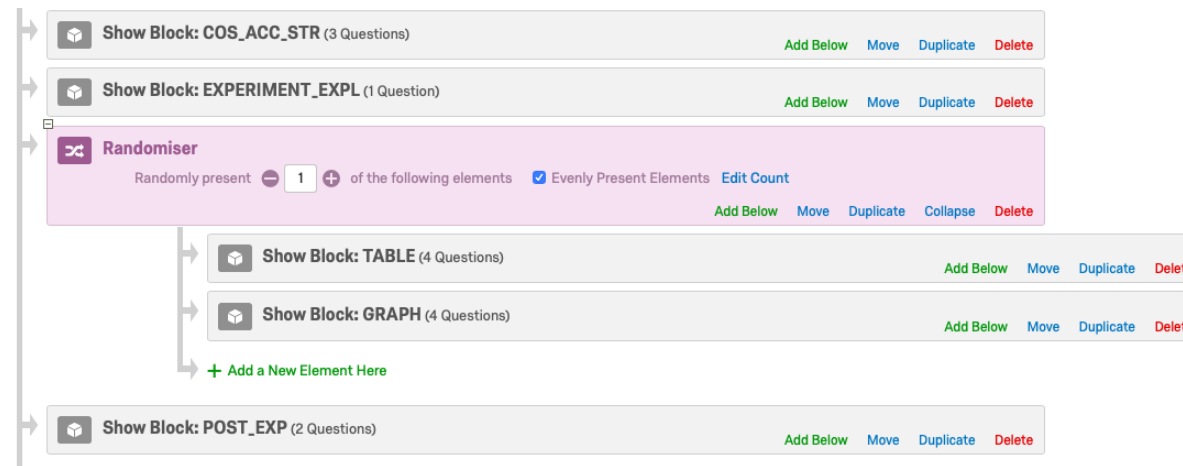
# Experimental Software



If you have a simple design, why would you make your life difficult?

Qualtrics is easy-to-use as it does not require any coding skills.

Especially useful if you are interested in judgments and decision-making.

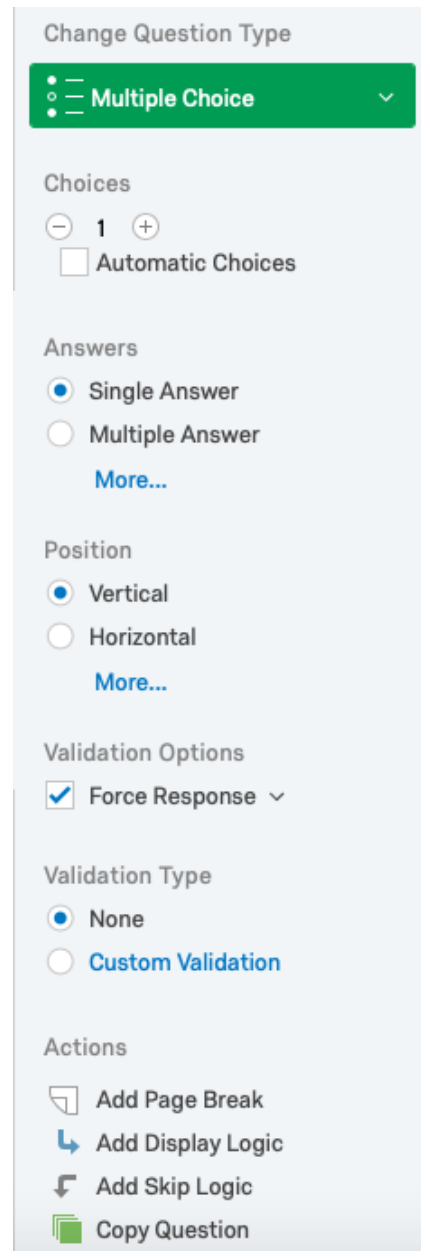


Lots of options → Easy GUI

Once you want more sophisticated / tailor-made options you get stuck

- Excellent for surveys or class experiments
- Excellent for single-participant or scenario-based experiments.
- Many tutorials online, does not take a long time to learn.

**Example:** A [Replication](#) of Cardinaels (2008) The interplay between cost accounting knowledge and presentation formats in cost-based decision-making *Accounting, Organizations, and Society*, 33(6), 582–602.



Change Question Type

• —  
• —  
• — Multiple Choice ▼

Choices

− 1 +

☐ Automatic Choices

Answers

☒ Single Answer

☐ Multiple Answer

[More...](#)

Position

☒ Vertical

☐ Horizontal

[More...](#)

Validation Options


☒ Force Response ▼


Validation Type


☒ None


☐ Custom Validation

Actions

 Add Page Break

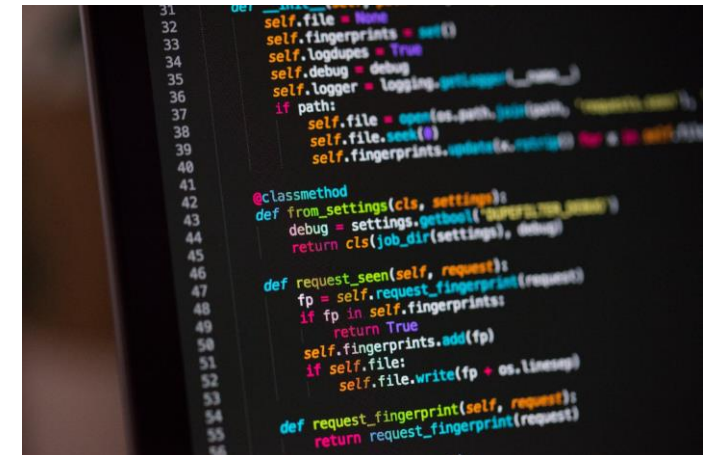
 Add Display Logic

 Add Skip Logic

 Copy Question

# oTree – Learning the Ropes

- **oTree** is an open-source platform that combines Python and Django (Chen, Schonger, and Wickens 2016).
- **Python** is an object-oriented programming language.
- **Django** is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.



```
31
32 self.file = None
33 self.fingerprints = self()
34 self.logdupes = True
35 self.debug = debug
36 self.logger = logging.getLogger(__name__)
37 if path:
38     self.file = open(os.path.join(path, 'requests.txt'),
39                       'a')
40     self.file.seek(0)
41     self.fingerprints.update(self.request.fingerprints)
42
43 @classmethod
44 def from_settings(cls, settings):
45     debug = settings.getboolean('DEBUG', False)
46     return cls(job_dir(settings), debug)
47
48 def request_seen(self, request):
49     fp = self.request_fingerprint(request)
50     if fp in self.fingerprints:
51         return True
52     self.fingerprints.add(fp)
53     if self.file:
54         self.file.write(fp + os.linesep)
55
56 def request_fingerprint(self, request):
57     return request_fingerprint(request)
```

# Why oTree?

- Platform-independent and online (not bound to the laboratory)
  - Participants do not need to install any software, just need internet browser
- Great visuals: everything can be adjusted with HTML, CSS, and JS.
- A lot of ready-made applications: e.g., public goods games, Keynesian beauty contest, Bomberman risk game
- Excellent control room for admin
  - Live-tracking
  - Automatic payment system



# Why oTree?

- Provides possibilities for more comprehensive research designs such as interactions, chat boxes, and integration with other software.
- For instance, you can let auditors do real audit tasks in an Excel spreadsheet (e.g, Dierynck and Peters, 2019)





DCF model used for the audit of the Feeder 2000 Patent

	B	C	D	E	F	G	H	I
2	<b>DISCOUNTED CASH FLOW ANALYSIS</b>							
3	<b>FEEDER 2000</b>							
4	<b>2018 Revenues of CMG Machines</b>	€ 40,000,000		<b>Growth Rates</b>				
5				Years 1-4	10%			
6	Discount rate	10%		Years 5-7	0%			
7	Royalty rate	10%						
8								
9	<b>Year</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>
10	Agrifirm	2019	2020	2021	2022	2023	2024	2025
11	Revenues of CMG Machines	€ 44,000,000	48,400,000	53,240,000	58,564,000	64,420,400	70,862,440	70,862,440
12								
13	Patent Royalty (10%)	€ 4,400,000	4,840,000	5,324,000	5,856,400	6,442,040	7,086,244	7,086,244
14								
15								
16	Present Value of CF from Royalty	€ 4,400,000	4,000,000	4,000,000	4,000,000	4,000,000	4,000,000	3,636,364
17								
18								
19	<b>Total Present Value of CF</b>	€ 28,036,364						



# Installation

You need the following software to program in oTree:

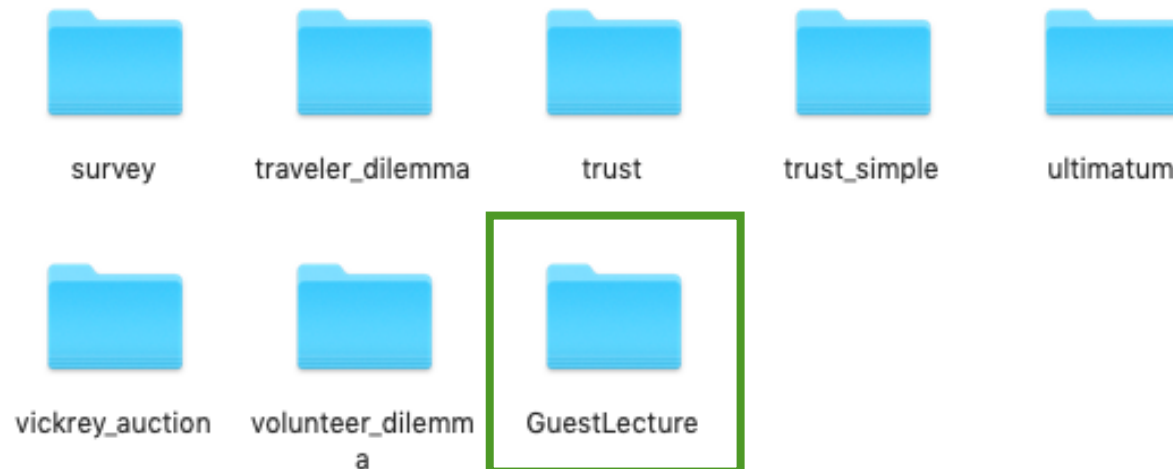
- Download Python 
- As integrated development environment (IDE) I use PyCharm. 
- Make sure you have some kind of version control, I use [GitHub](https://github.com). 
  - If you crash your code, you can always go back to a previous version of the code.
- Install oTree with Python in the command prompt. 

# Structure

- If you install oTree, it shows up as a folder on your computer (e.g., /Users/Christian/oTree)
- In this folder there are many applications: ready-made apps.
- You can also create your own app:

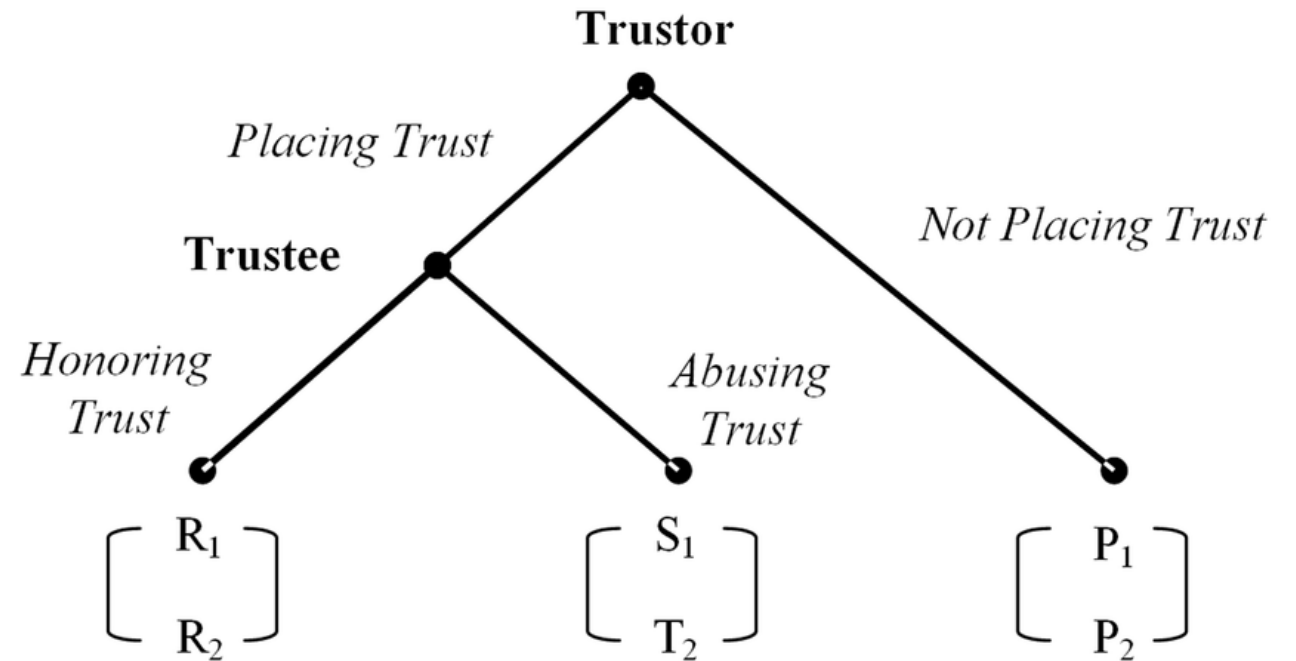
*cd 'oTree Folder'*

*otree startapp GuestLecture*



# An Example

- An example experiment is the [Trust Game](#) (Berg, Dickhaut, and McCabe, 1995)
- One **trustor**, one **trustee**.
- **Multiplication factor of 3**
- An initial **endowment** for the trustor



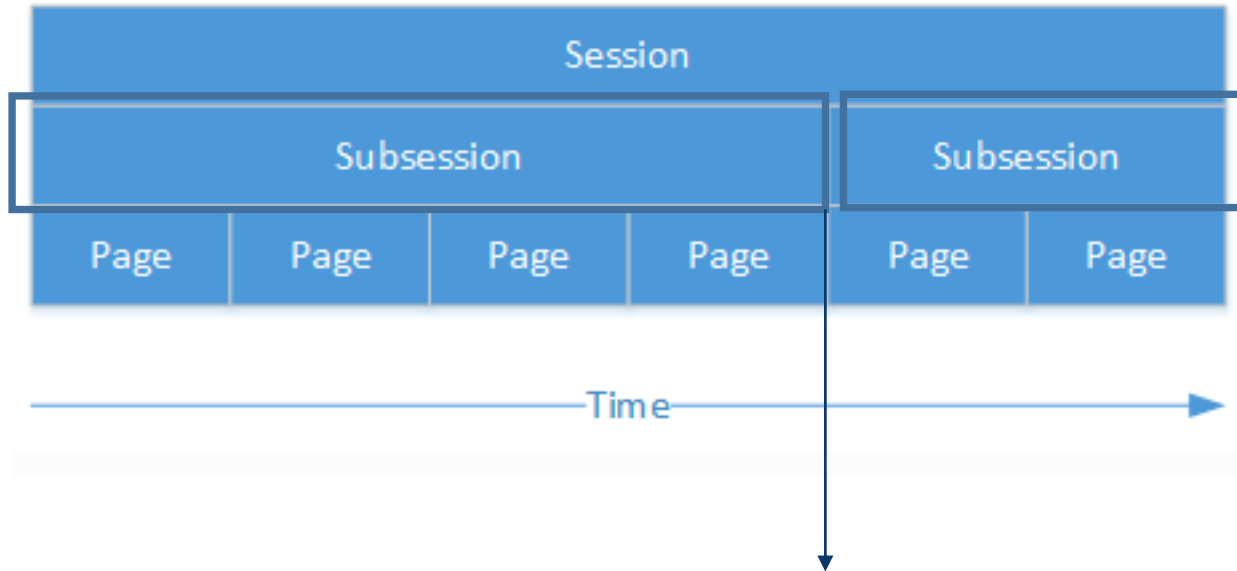
$$R_i > P_i (i = 1, 2), P_1 > S_1, T_2 > R_2$$

# The Four Building Blocks

- **Templates**
  - Every page in your experiment is a template with **text** and **layout**.
- **Models.py**
  - Used to define **every variable** that you use in your experiment.
- **Pages.py**
  - Used to determine **what** is showed, **when**, and **how long**?
- **Settings.py**
  - Used to determine **payoff structure** and general application settings

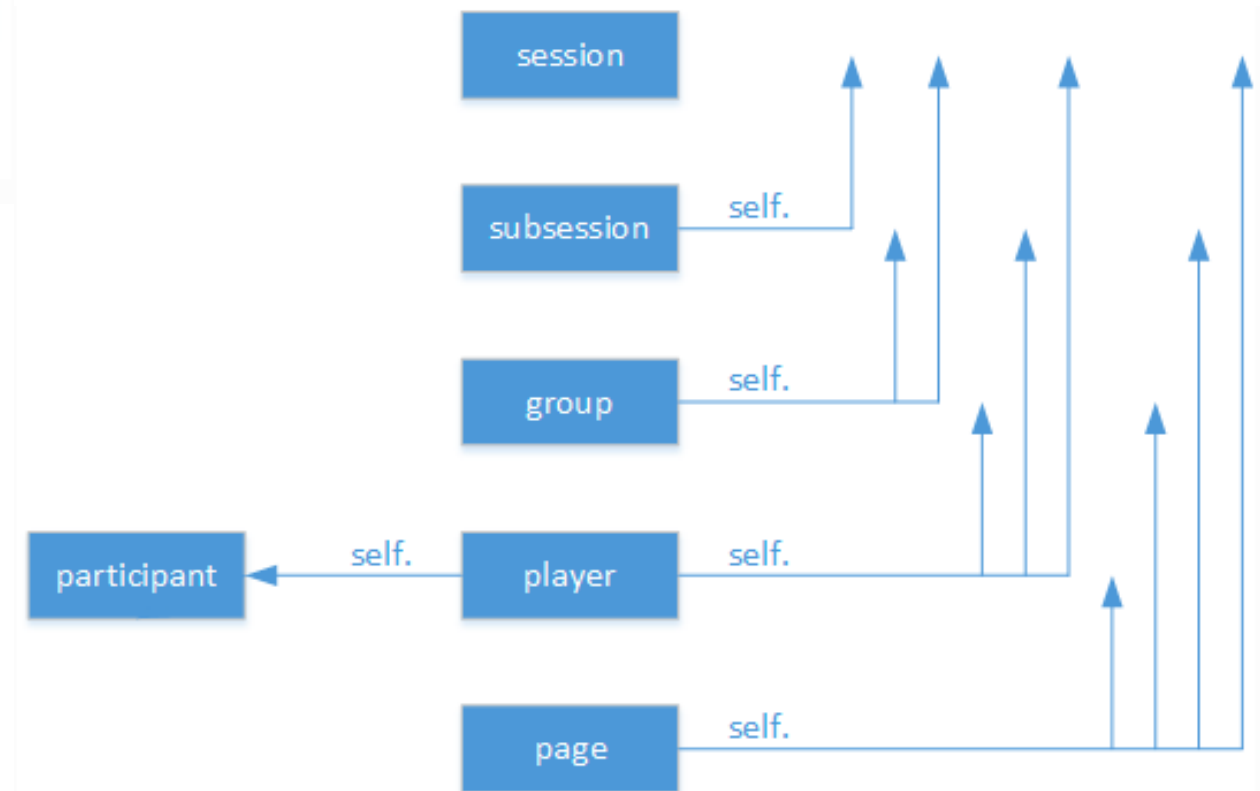


# Models.py (Structure)



Subsessions can for example be a trust game, an ultimatum game, or a survey.

## Classes



# Models.py

- **Models.py** is used to create variables that you use in your experiment.
- Some things are constant in the trust game and we want to specify those in the **Constants class**.

- An endowment
- A multiplier
- The number of rounds
- The number of players in each round

```
class Constants(BaseConstants):  
    namein_url = 'example_trust'  
    players_per_group = 2  
    num_rounds = 1  
  
    instructions_template = 'trust/Instructions.html'  
  
    # Initial amount allocated to each player  
    endowment = c(10)  
    multiplier = 3
```

# Models.py

- Suppose you want data like this for a trust game:

Group	sent_amount	sent_back_amount
adj523ld	€3	€6
19hupfey	€5	€0
enoordo8	€4	€10

```
class Group(BaseGroup):
    sent_amount = models.CurrencyField(
        choices=currency_range(0, Constants.endowment, c(1)),
        doc="""Amount sent by P1""",
    )

    sent_back_amount = models.CurrencyField(
        doc="""Amount sent back by P2""",
    )
```

**Constants**  
“global” variables  
e.g. multiplier = 3

**Fields**  
StringField  
LongStringField  
IntegerField  
BooleanField  
FloatField  
CurrencyField

Understanding Society

# Templates

- For every page, you need to make a template: this is what participants see.
- You can use HTML, CSS, Django, and Javascript.
- Django tags are especially useful.

You can input variables like this

```
<h3>
```

Instructions

```
</h3>
```

```
<p>
```

This is a trust game with 2 players.

```
</p>
```

```
<p>
```

To start, participant A receives {{ Constants.endowment }}  
participant B receives nothing.

Participant A can send some or all of his {{  
Constants.endowment }} to participant B.

Before B receives this amount it will be tripled.

Once B receives the tripled amount he can decide to send  
some or all of it back to A.

```
</p>
```



# Templates

- You can also condition in templates. For instance, if you have a textual manipulation.

{% if player.treat in Constants.odd %}

In this audit team you had **a lot of** control over the way in which you conducted the previous audit tasks. Therefore, you **can** use the insights from the previous tasks in deciding how to conduct future audit tasks.

{% else %}

In this audit team you had **no** control over the way in which you conducted the previous audit tasks. Therefore, you **cannot** use the insights from the previous tasks in deciding how to conduct future audit tasks.

{% endif %}

# Pages.py

P1: Send

P2: Send Back

Results

- We also need to create waitpages.

```
page_sequence = [  
    Send,  
    WaitForP1,  
    SendBack,  
    ResultsWaitPage,  
    Results,  
]
```

```
class Send(Page):
```

```
    form_model = 'group'  
    form_fields = ['sent_amount']
```

```
    def is_displayed(self):  
        return self.player.id_in_group == 1
```

```
class WaitForP1(WaitPage):  
    pass
```

# Experimental Settings (Settings.py)

- Here, you determine the general settings of your experiment:

```
SESSION_CONFIG_DEFAULTS = {  
    'real_world_currency_per_point': 1,  
    'participation_fee': 1,  
    'doc': "",  
}  
  
SESSION_CONFIGS = [  
    {  
        'name': 'trust_behavioral',  
        'display_name': 'Trust Game Behavioral Analysis',  
        'num_demo_participants': 2,  
        'app_sequence': ['trust_simple'],  
    },  
]
```

# Making your Own Online Experiment

- Now we have every ingredient to make our own online experiment.

Models.py

Pages.py

Settings.py

Templates



# It's Ready!

## Trust Game: Your Choice

### Instructions

This is a trust game with 2 players.

To start, participant A receives 10 points; participant B receives nothing. Participant A can send some or all of his 10 points to participant B. Before B receives this amount it will be tripled. Once B receives the tripled amount he can decide to send some or all of it back to A.

You are Participant A. Now you have 10 points.

How much do you want to send to participant B?

5 points

Next



# A Live Example

- Let's try to make a relatively easy experiment to test **attribute substitution**.
- Strack, Martin, and Schwartz (1988) asked participants two questions:
  - “How happy are you with your life in general?”
  - “How many dates did you have last month?”
- When asked in this order, the correlation was negligible. However, when the dating question was asked first, correlation rose to .66.
- Participants substituted the complex question of how happy they are with the easier (heuristic attribute) of how many dates they had last month.

# A Live Example

- A simple experiment: only two questions and two treatments (order)
- We need only three variables at the player-level in **Models.py**:  
treatment = models.IntegerField()  
question\_1 = models.IntegerField(label="", blank=False)  
question\_2 = models.IntegerField(label="", blank=False)
- For treatment, we need a function to assign a participant to a treatment (at the subsession-level):  
def creating\_session(self):  
 import itertools  
 number = itertools.cycle([0,1])  
 for player in self.get\_players():  
 player.treatment = next(number)

# A Live Example

- In **Pages.py** we need to create two pages, define what variables are asked on which page and determine the page sequence.

```
class Page1(Page):  
    form_model = 'player'  
    form_fields = ['question_1']
```

```
class Page2(Page):  
    form_model = 'player'  
    form_fields = ['question_2']
```

```
page_sequence = [Page1, Page2]
```



# A Live Example

- We need to create **Templates** for the pages:

```
{% if player.treatment == 0 %}
```

```
    <p>How many friends do you approximately have on social media?</p>
```

```
{% else %}
```

```
    <p>Please indicate on a scale from 1 to 10 how happy you are with your life in  
    general.</p>
```

```
{% endif %}
```

```
{% formfields %}
```



# HEROKU

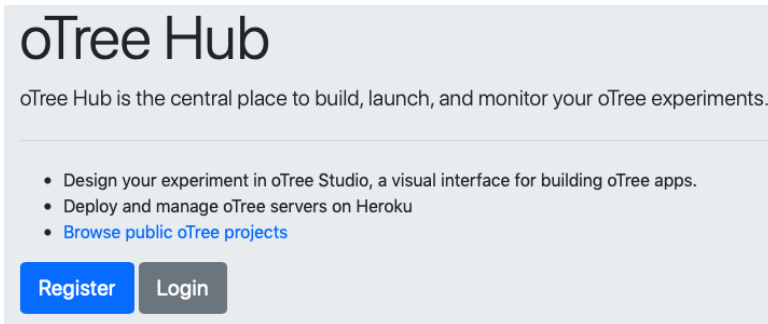
## Server



- In order to make the experiment available to others, we need a place to store the experiment.
- I use an online server called **Heroku**.
- Heroku is a cloud platform as a service supporting several programming languages.
- You **pay** for the server capacity you **use**. If you use a server that costs \$100 per month for one day, you pay only \$3.33.



- There are **three ways** to deploy your experiment to the Heroku server.



- Easy, no coding necessary.
- Recommended if not familiar with coding in the command prompt and git.
- Disadvantage: pay or public



- Integrate with GitHub
  - Automatically
  - Easy if you have GitHub



- Everything in your own control
  - But you need to code!
  - The exact code needed can be found in the repository of this presentation at <http://github.com/CPHPeters>  
**Understanding Society**



Now the experiment is on a server, let's try our online experiment!

Visit: <http://christianpeters.herokuapp.com>

<a href="#">Description</a> <a href="#">Links</a> <a href="#">Edit</a> <a href="#">Monitor</a> <a href="#">Data</a> <a href="#">Payments</a> <a href="#">MTurk</a>								
ID in session	Code	Label	Page	App	Round	Page name	Status	Time on page
P1	2xf9u6wc		10/9 pages	publictax	1	Turk	Playing	5 days ago
P2	0qn0qc1g		10/9 pages	publictax	1	Turk	Playing	5 days ago
P3	iz6lpb46		1/9 pages	publictax	1	Intro	Playing	5 days ago
P4	slbn9o3q		1/9 pages	publictax	1	Intro	Playing	5 days ago
P5	pxodlnff		10/9 pages	publictax	1	Turk	Playing	5 days ago
P6	2rsyo1y5		10/9 pages	publictax	1	Turk	Playing	5 days ago
P7	ubmrX06l		10/9 pages	publictax	1	Turk	Playing	5 days ago
P8	alxytzz		10/9 pages	publictax	1	Turk	Playing	5 days ago
P9	c8uqx9ca		10/9 pages	publictax	1	Turk	Playing	5 days ago
P10	c4wodn7y		10/9 pages	publictax	1	Turk	Playing	5 days ago

# Online Worker Pools

- Now we have the experiment on an online server, we want to deploy it to many participants. There are **two options**:



- Easy-to-use
- Often low-quality participants
- Cheap
- Seamless integration with oTree



- More difficult to use
- Higher quality participants
- More expensive
- More effort to set up the experiment

# Amazon Mechanical Turk

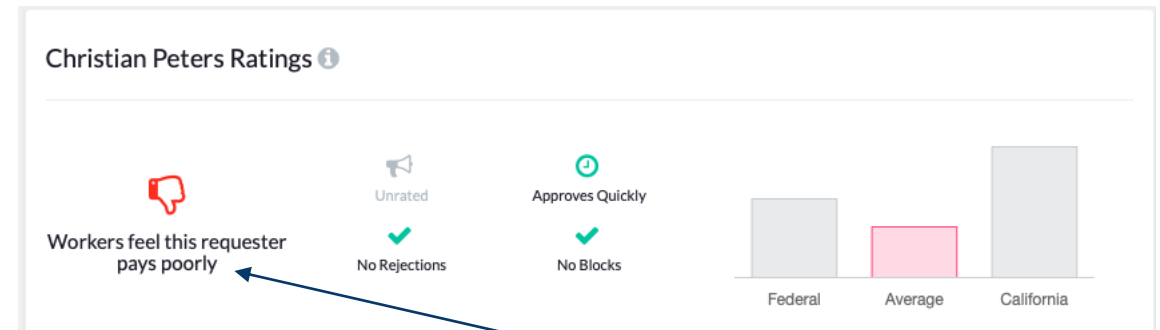
- Farrell, Grenier, and Leiby (2017, TAR): “*Collectively, we provide evidence of high-quality, low-cost source of work on demanding tasks that are characteristic of accounting research.*”
- The paper suggests that M-Turkers are stars. I think their conclusion should come with a caveat. In my experience, they are stars but only for:
  - ❖ Easy Tasks
  - ❖ Sentiment Analysis
  - ❖ Pre-Tests
- And not for:
  - ❖ Complex Tasks
  - ❖ Lengthy Tasks
  - ❖ Interactive Tasks



# A M-Turk Manual

For your Human Intelligence Task (HIT) you need:

- Impeccable customer service
- Minimize reading
- Minimize time
- High pay
- Clear and concise instructions
- Indication of time and pay in title/description
- Customary to include feedback box



*“Do as I say, don’t do as I do”*

# An Active Community

- Who are M-Turkers?

well this was a great morning. I'm not sure if i will be back or not tonight. depends on if the toddler wants to play

Projected Earnings \$39.41

Have a great afternoon!!

💬 Originally Posted by **DarlsBored** 📺

*This morning was soooo good and now I'm struggling to put together a dollar*

Agreed, I almost made 60.00 by noon, now I have taken the dog for a walk, packed lunches, cleaned etc, and not a single panda is ringing



# An Active Community

- M-Turkers are an active community (e.g., TurkerNation)
- It is important to provide impeccable customer service to Turkers → They actively respond.

#240

Originally Posted by mturkmaster


Ties de Kok  
Classify 5 tweets directed at game developers.  
  
all rejected, i knew they were too good to be true. avoid at all costs. these were easy.

Report requester and not necessarily because all were rejected but the name is offensive. IDK if you even can report a requester and not just the hit itself, never had to luckily (many times I should have probably). Unless it's Ties de Kok a PhD researcher at Tilburg University that specializes in combining computer science with empirical Accounting research.

FU CHEAPSKATE

LMAO. YAH OK. ALL FOUR OF THEM WERE COMPLETELY WRONG. I'M SURE. THIS WASN'T ROCKET SCIENCE, AND I WAS PRETTY DAMN FUCKING CERTAIN I HAD THEM RIGHT, SCREW OFF. BAD TO COMING.

GimmeHits said: ↑  
Tease de who ?



TILBURG UNIVERSITY

Understanding Society

# How to Link your Experiment with M-Turk

1. Create a “HIT” on Amazon’s Mechanical Turk, where you provide the link to your Qualtrics survey.
2. Use a random number generator in Qualtrics to assign a random number to every participant at the end of the experiment.
3. Turkers enter this randomly generated code into M-Turk.
4. Verify that each entered code in M-Turk is assigned by Qualtrics.

# How to Link your Experiment with M-Turk

- Integrating oTree with M-Turk is very easy.
- You can use qualification requirements to get only certain participants (e.g., U.S. only)
- You need to set CONFIG VARS in Heroku ([Click](#)).
- Then, you are able to run the experiment on M-Turk
  - Payment is calculated automatically.

Create new session

For MTurk

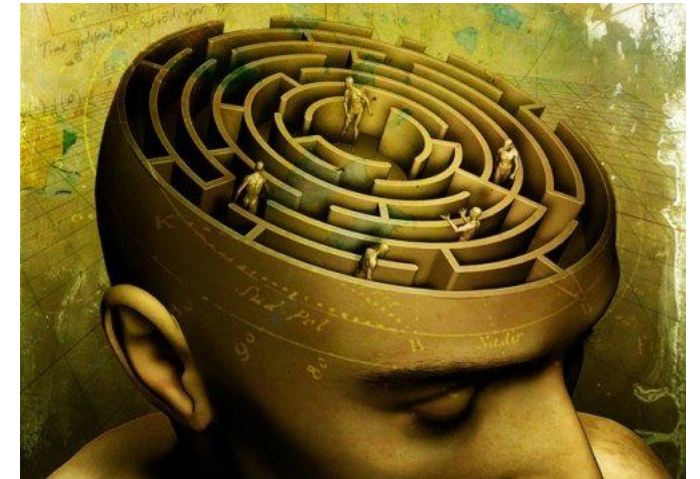
# A note on Interaction and M-Turk

- It is difficult to conduct interactive experiments on M-Turk.
  - Participants start at different points in time.
  - Participants might quit during the experiment.
  - Participants may do something else (play with a toddler, Facebook)
- A way to solve this is the **strategy method** (we elicit full strategies).

*What would you send?*

*What would you send back if you received \$1, \$2, ..., \$10?*

Match Participants





# Project Management

- Coding can be difficult and mistakes can easily be made.
- Sometimes a mistake is really hard to solve. I make a lot of mistakes.
- **My tip:** Use version control: best platform is GitHub.
  - This saved me a couple of times!

You can find a repository for this guest lecture on: [github.com/CPHPeters](https://github.com/CPHPeters)

Overview **Repositories 8** Projects 0 Packages 0 Stars 1 Followers 1 Following 6

Find a repository... Type: All Language: All **New**

**Guest\_Lecture\_Experiments** ★ Star  
Repository for a Guest Lecture for Behavioral Analysis of Accounting at Tilburg University.  
MIT License Updated 2 days ago

**Data** Private ★ Star  
Data for Research Projects  
Updated 5 days ago

★ PRO

# Building your own Experiment is an Art

“Black Square”, Kazimir Malevich, 1915

X-Ray reveals: there are two layers of complete paintings below the surface

Some believe the cracks in the black are deliberate. The underlying paintings start to show through over time

Black square is the **end** of a long and hard thought process

**Don't forget to enjoy** building your own experiment!



# References

- Berg, J., Dickhaut, J., & McCabe, K. (1995). Trust, Reciprocity, and Social History. *Games and Economic Behavior*, 10(1), 122-142.
- Cardinaels, E. (2008). The interplay between cost accounting knowledge and presentation formats in cost-based decision-making. *Accounting, Organizations and Society*, 33(6), 582-602.
- 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.
- Dierynck, B. and Peters, C.P.H. (2019). Auditor Task Selection under Time Pressure. *Working Paper*. Available at SSRN: <https://ssrn.com/abstract=3450363>
- Farrell, A. M., Grenier, J. H., & Leiby, J. (2016). Scoundrels or stars? Theory and evidence on the quality of workers in online labor markets. *The Accounting Review*, 92(1), 93-114.
- Fischbacher, U. (2007). z-Tree: Zurich toolbox for ready-made economic experiments. *Experimental Economics*, 10(2), 171-178.
- Strack, F., Martin, L. L., & Schwarz, N. (1988). Priming and communication: Social determinants of information use in judgments of life satisfaction. *European Journal of Social Psychology*, 18(5), 429-442.