CREED Experimenting Guide

Updated May 2025
Tell me if anything is wrong or missing and we can update it!

Contents

1 Experimental Timeline

1.1 Overview

Before running an experiment at the CREED Lab or applying for ACBC funding, present your project at the CREED Lunch Seminar. Contact Julian Kirschner or Madgalena Wasilewska to schedule. Complete the following steps:

1. **Pre-analysis plan:** AsPredicted

2. **Funding application:** ACBC

3. **UvA/EB Project Overview:** Research Management Services

   - Ethics review
   - Data Management Plan

Pre-Analysis Plan Questions.

1. Has any data already been collected?

2. What is the main question or hypothesis?

3. Key dependent variable(s) and measurement method?

4. Number and nature of conditions?

5. Planned analyses?

6. Outlier handling and exclusion criteria?

7. Sample size or rule?

8. Additional info?

2 Running Experiments in the Lab

We use the CREED website to manage everything about participant sign-ups. In the lab you run your experiment from your oTree folder on the local server. You need to make sure you i) reserve the lab, ii) get the participants invited, and iii) your code works on all the computers in the lab. Find instructions on how to make sure this all works below.

2.1 Reserving the Lab

Coordinate with Joep to reserve lab access and receive credentials for CREEDExperiment.nl.

2.2 CREED Website - Managing Participant Sign-up

Getting Started with the Website.

- Login details: from Joep

- Experiment number: from Ailko

- URL: `http://www.creedexperiment.nl/proftulp/loginexp.html`

- Create a new session via "new exp"

Managing Experimental Sessions.

- Modify session: `Show Session > Modify`

- Send reminder: `Show Session > Reminder`
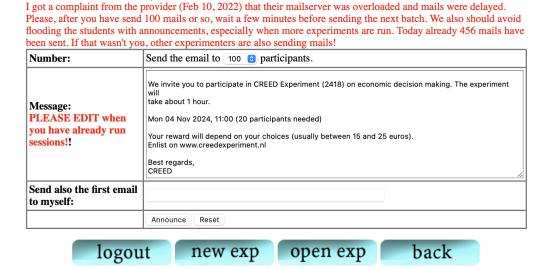
- Finish session: `Show Session > Finish`

Inviting Participants.

1. click 'make a table with potential subjects with name and email address'

2. click 'send emails to a limited number of subjects (like 100)'

3. Prepare the customisable email template

4. Limit email sends (e.g., 100 per batch)

5. System tracks sent emails and removes people from the table to avoid re-sending to the same participant.
   Only create new tables the first time you send invites or when you don't have enough 'mails to send' in the screen attached below. Once you create a table it removes anyone you have already sent a mail to from the table, hence why the number decreases. This avoids sending duplicates.

You have still **295** mails that you can send.

I got a complaint from the provider (Feb 10, 2022) that their mailserver was overloaded and mails were delayed. Please, after you have send 100 mails or so, wait a few minutes before sending the next batch. We also should avoid flooding the students with announcements, especially when more experiments are run. Today already 456 mails have been sent. If that wasn't you, other experimenters are also sending mails!

Number:	Send the email to ⟨100⟩ participants.
Message: **PLEASE EDIT when you have already run sessions!!**	We invite you to participate in CREED Experiment (2418) on economic decision making. The experiment will take about 1 hour. Mon 04 Nov 2024, 11:00 (20 participants needed) Your reward will depend on your choices (usually between 15 and 25 euros). Enlist on www.creedexperiment.nl Best regards, CREED
Send also the first email to myself:	
	Announce Reset

logout   new exp   open exp   back

Sending Reminders. Navigate to Show Sessions, select a session, and click `Reminder`.

- You can customise the reminder (subject, message).

- Add your email to test delivery.

- Double-check it's for the right session before sending.

2.3   Running in the Lab

Here are instructions to implement a super simple workflow when you enter the lab. All you have to do to start it up and monitor the session then is:

1. turn on all the computers

2. run experiment.bat on the experimenter computer

3. open shortcut 'Chrome to 'experiment' Room Small/Large Lab' on the participant desktop

4. confirm your session configs in the oTree monitoring page

Configuration of oTree settings.py. Insert this code into your `settings.py` file. Ensure you don't have other rooms and don't hard-code the ADMIN_USERNAME or ADMIN_PASSWORD elsewhere.

```python
ROOMS = [
    dict(
        name='experiment',
        display_name='Experimental Session',
    ),
]

# Admin credentials from environment
import os
ADMIN_USERNAME = os.environ.get('OTREE_ADMIN_USERNAME')
ADMIN_PASSWORD = os.environ.get('OTREE_ADMIN_PASSWORD')

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': os.environ.get('DB_NAME'),
        'USER': os.environ.get('DB_USER'),
        'PASSWORD': os.environ.get('DB_PASSWORD'),
        'HOST': os.environ.get('DB_HOST', 'localhost'),
        'PORT': os.environ.get('DB_PORT', '5432'),
    }
}
```

Listing 1: settings.py configuration

Displaying Participant Computer Terminal as Player Labels. Insert this code into the __init__.py file of your first app.

```python
class Player(BasePlayer):
    participant_label = models.StringField()

    def set_participant_label(self):
        self.participant_label = self.participant.label
```

Listing 2: First app init.py

Start Session Script. Use this '.bat' script to automate the server and environment start-up:

```bat
@echo off
REM === Database Setup ===
REM psql -U postgres
REM CREATE DATABASE experimentname;
REM CREATE USER moi WITH PASSWORD '1234';
REM GRANT ALL PRIVILEGES ON DATABASE experimentname TO moi;

REM === Database Management ===
set DB_NAME=experimenting
set DB_USER=moi
set DB_PASSWORD=ExperimentAllDay
set DB_HOST=localhost
set DB_PORT=5432
set DATABASE_URL=postgres://%DB_USER%:%DB_PASSWORD%@%DB_HOST%:%DB_PORT%/%DB_NAME%

REM === OTree variables ===
set OTREE_ADMIN_USERNAME=admin
set OTREE_ADMIN_PASSWORD=1234
set OTREE_PRODUCTION=1
set OTREE_AUTH_LEVEL=STUDY

REM === Starting the OTree project up on the server ===

REM === !!! ADOPT YOUR PATH !!! ===
cd "C:\Users\Admin\Desktop\TAIT\ExpMistakes Copy"
otree resetdb

REM === !!! SELECT LAB !!! ===

REM SMALL LAB: 145.18.178.133
REM LARGE LAB: 145.18.178.130

start http://145.18.178.133:8000/rooms

echo DON'T FORGET TO SELECT THE RIGHT LAB!!!

otree prodserver

echo Your postgres server is up and running
```

Listing 3: start_experiment.bat

Monitor the Session. experiment.bat will automatically open the monitoring page. If not, visit `http://145.18.178.130:8000/rooms` for the large lab or `http://145.18.178.133:8000/rooms` for the small lab.

2.4 PostgreSQL Setup

If you follow the steps above start_experiment.bat you will automatically run with **PostgreSQL**. If not, you should know that oTree defaults to a basic database called SQLite. This can lag more and does not back up your data in case the computer crashes. I recommend you ensure you're using PostgreSQL when running sessions to avoid problems.

If you download your session data after each session and are happy with other wiping the database after every session (ensuring you already have it saved, see 3) you can use an existing PostgreSQL

database to read and write your data throughout the session (this is already specified to start in start_experiment.bat). Alternatively, you can also create your own database using i) the code below, or ii) simply opening 'pgAdmin 4' on the experimenter computer and following the steps below.

Using *pgAdmin 4*. This is the simplest way with a visual interface.

```
1.  Create a new user and password:
    - Open pgAdmin.
    - Right-click Login/Group Roles $\rightarrow$ Create $\rightarrow$ Login/Group Role.
    - Set:
    - Name: newname
    - Password tab: enter newpassword
    - Privileges tab: check all boxes`

2.  Create the database:
    - Right-click Databases $\rightarrow$ Create $\rightarrow$ Database
    - Name it experimentname
    - Set owner to newname
```

Using the terminal. The master password should be '1234'.

```
psql -U postgres
CREATE DATABASE experimentname;
CREATE USER moi WITH PASSWORD '1234';
GRANT ALL PRIVILEGES ON DATABASE experimentname TO moi;
```

Then set the 'DATABASE_URL' environment variable accordingly in the start_experiment.bat script.

3 Managing Lab Session Data

Download it from oTree after each session is completed in the 'data' tab. Save it in some data folder in your project folder as 'Session_XX.csv'. This ensures that whatever happens to the database on the experimenter computer you already have your data elsewhere.

You can then compile this together for your overall dataset.

4 Paying Participants

4.1 Cash

Ask your supervisor to email Ailko (ailko@xs4all.nl) with:

- Experiment number
- Total cash needed
- Preferred denominations

Bring them up one-by-one to pay them the cash in the correct amounts. Make sure they sign a receipt of payment so you can prove where the cash has gone.

4.2 Digital Transfer (UvA Administration)

Overview: Participants can be paid via bank transfer from the central UvA administration (SEPA only). If their IBAN is not in the SEPA region, arrange an alternative (cash to be collected from your office).

1. **Collect Payment Details**

   - Collect **IBAN** from all participants.

   - If IBAN is not Dutch, also collect the **BIC/SWIFT** code.

   - Check if IBAN belongs to a SEPA country.

   - If not SEPA, show: *"Please speak to the experimenter before leaving."*

2. **Handle Non-SEPA Participants**

   - Inform them that **cash payment** will be arranged later.

   - Record their contact details and agree on an office pickup time.

   - You can enter your bank details into the payment schedule and source the cash privately. This proved a very small share of total participants (6 of 350).

3. **Submit SEPA Bank Payments**

   - Compile valid IBANs in the required format (download link)

   - Email `servicedesk-ac@uva.nl` with the payment list and request processing.

4. **Track Payment Status**

   - Look for two emails: *confirmation of processing* and *confirmation of payment*.

   - Keep a log of any **rejected IBANs** and the **session** that participant was in so you can identify people who reach out to CREED.

5 FAQ

5.1 oTree/Heroku

Q: How many participants can Heroku handle at once?. Based on experience, standard setups can support around **20–50 participants** simultaneously. Going beyond this increases the risk of server crashes.

Q: How can I scale to more participants (e.g., 80–120)?. A setup that successfully handled up to 120 participants used the following Heroku configuration:

- **Heroku Postgres Standard 0** ($50/month)

- **Heroku Redis Premium 2** ($60/month)

- **Heroku Dynos:**

  - Web dynos: 2X with 4 dynos ($200/month)

  - Worker dynos: 2X with 2 dynos ($100/month)

Tip: Dynos and Redis can be scaled up before a session and down afterward to reduce costs. However, the Postgres database must remain active to retain session data.

Q: Wait pages don't refresh automatically. What should I do?. If you encounter this issue, make sure to run:

```
heroku config:set PGSSLMODE=require
```

Listing 4: Heroku SSL Configuration

Without this, you may get the error: `FATAL: no pg_hba.conf entry for host`, and wait pages won't auto-update.

Q: Is there a workaround to prevent stuck wait pages?. Yes. You can add a fallback script that reloads the page every 10 seconds to avoid session breakdowns:

```
setInterval(function(){ location.reload(); }, 10000);
```

Listing 5: Auto-refresh fallback

Note: This helped recover from issues during high-load sessions where the auto-update failed.