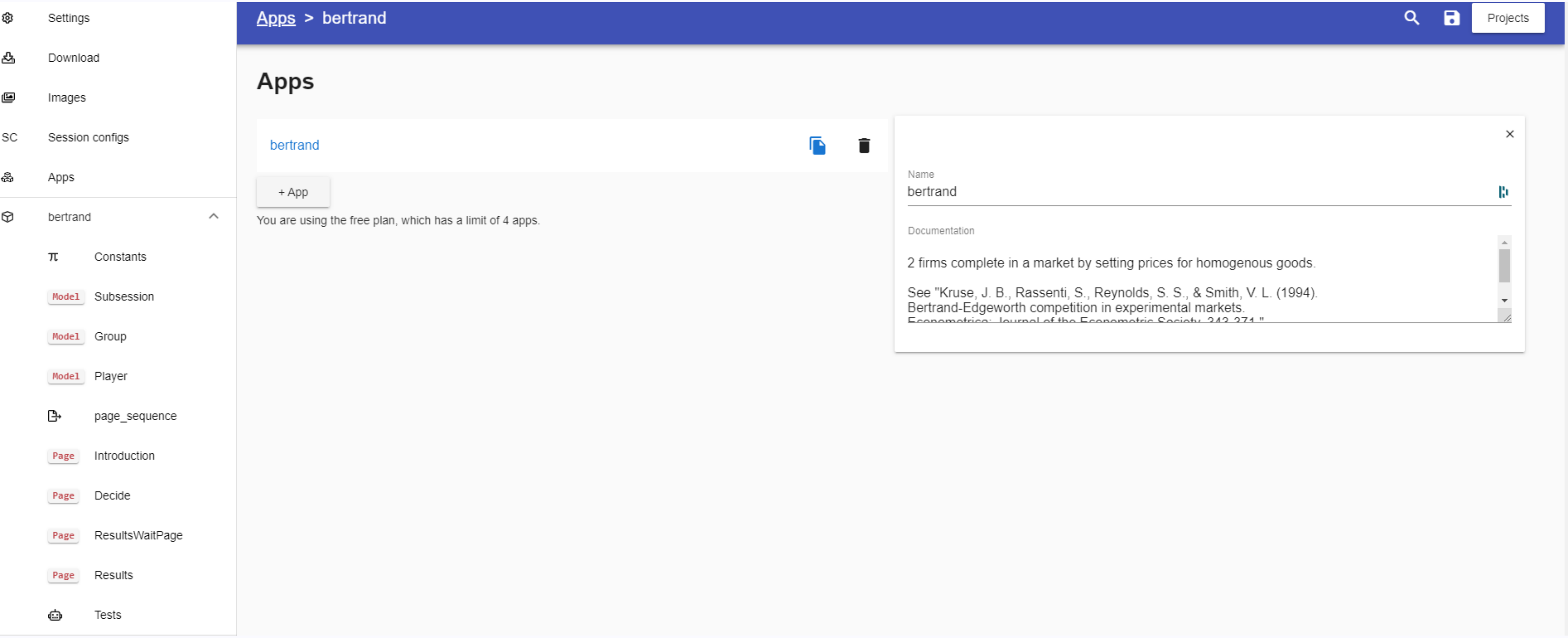
All the rest

Remixed from material by Ali Seyhun Saral & Philipp Chapkovski

oTree Studio



Apps

bertrand

You are using the free plan, which has a limit of 4 apps.

Name
bertrand

Documentation

2 firms complete in a market by setting prices for homogenous goods.

See "Kruse, J. B., Rassenti, S., Reynolds, S. S., & Smith, V. L. (1994).
Bertrand-Edgeworth competition in experimental markets.

oTree Studio

- Web-based tool with a visual interface for building oTree apps
- Tries to be more user-friendly
- To access it visit otreehub.com

oTree Lite

oTree lite:

- oTree that runs as a self-contained framework, not dependent on Django
- Promise to be simpler and more lightweight
- Code base is more self-contained

Reason to use it:

- (Probably) easier to start as first-time otree user
- Easy to change code from existing apps if those have been developed more recently (last years)

Reason to avoid it:

- Your apps are complex and/or build on existing apps
- You use Django features

oTree Lite: Examples

```
class Group(BaseGroup):
    unit_price = models.CurrencyField()
    total_units = models.IntegerField(doc="""Total units produced by all players""")

class Player(BasePlayer):
    units = models.IntegerField(
        min=0,
        max=Constants.max_units_per_player,
        doc="""Quantity of units to produce""",
        label="How many units will you produce (from 0 to 30)?",
    )

# FUNCTIONS
def set_payoffs(group: Group):
    players = group.get_players()
    group.total_units = sum([p.units for p in players])
    group.unit_price = Constants.total_capacity - group.total_units
    for p in players:
        p.payoff = group.unit_price * p.units

def other_player(player: Player):
    return player.get_others_in_group()[0]
```

oTree Lite

<u>Personal hot takes on oTree lite:</u>

- oTree developers say that both version oTree lite/oTree will be developed in the future
- My guess is that in the mid/long run otree lite will become otree
- Less object oriented, more functional
  - Probably easier to learn
  - You lose the power that Django offers

Testing in zTree

<u>zTree testing "strategies":</u>

- Open a bunch of zleaf programs and do it manually
- Invite your co-workers to the lab and let them click through the experiment (BTW: It's not fun)

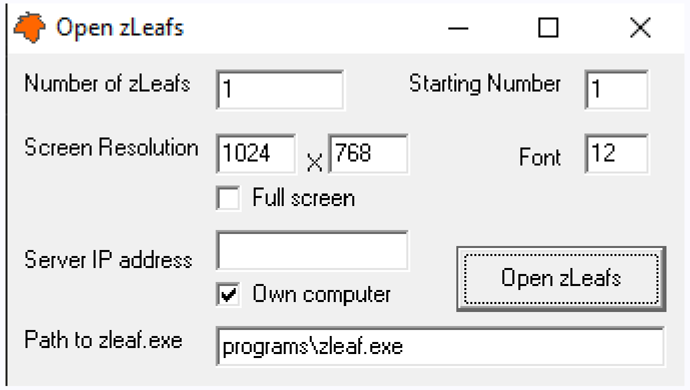
<u>So many problems with that:</u>

- Error-prone as people usually do not do it strategically
- Take time and is inefficient
- Testing matching schemes becomes difficult
- …

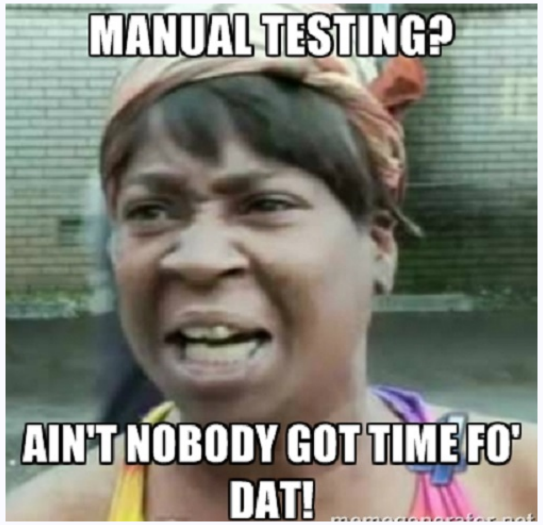
<u>Result:</u>

- People test too little
- You notice bugs too late

Q: How many of you still found errors in your zTree code after the first pilots?


Open zLeafs

Number of zLeafs	1	Starting Number	1
Screen Resolution	1024 x 768	Font	12
	☐ Full screen		
Server IP address			
	☑ Own computer		
Path to zleaf.exe	programs\zleaf.exe		



Testing in oTree

<u>oTree testing strategies:</u>

- Write automate tests (Bots) to check your code
    - Bots.py file in your app folder
- Run the tests from the command line or directly in the browser
- Also possible to test your code directly on the webserver in case you do an online experiment

<u>oTree tests have a lot of flexibility:</u>

- Bots plays round and you tell him what to expect on each page
- You can submit pages with formfield input
- Differ cases (Cooperative/Noncooperative types)
- Check your matching
- Check if timeouts work
- Export your data to have some fake data to already play around with and prepare analysis scripts

Testing in oTree: Example Public Goods Game

```
from . import pages
from otree.api import Bot, SubmissionMustFail


    class PlayerBot(Bot):

      cases = ['basic', 'min', 'max']

      def play_round(self):
      yield (pages.Introduction)
          contribution = {
            'min': 0,
            'max': 100,
            'basic': 50,
          }[self.case]

        yield (pages.Contribute, {"contribution": contribution})
      yield (pages.Results)

        if self.player.id_in_group == 1:

           if self.case == 'min':
               expected_payoff = 110
            elif self.case == 'max':
               expected_payoff = 190
            else:
               expected_payoff = 150
            assert self.player.payoff == expected_payoff
```

Deployment & Running your experiment

Offline lab experiments:

- Simple process in our lab that use the oTree Room feature
- Documentation in the wiki: https://wiki.hhu.de/display/dicelab/How+to+oTree
  - Tell Gerhard if you are not part of the wiki yet

Online experiments:

- Online lab by DICE: onlinelab.dice.hhu.de
- oTreehub.com
- Heroku: https://github.com/oTree-org/otree-docs/blob/143a6ab7b61d54ec2be1a8bc09515d78e0b07c71/source/server/heroku.rst#heroku-setup-option-2
- Additional features if you want to use Amazon Mturk: https://otree.readthedocs.io/en/self/mturk.html

Live pages & Dynamic pages

Live pages

- Live pages communicate with the server continuously
- Used for continuous time games
- Games with lots of fast iteration and interaction between users
- Need to program in Javascript

Note on Javascript & oTree:

- Powerful programming language to make your experiment interactive
- Code is executed on the client-side
- Offers a huge potential for creating dynamic experiments
- Use it if it makes sense, avoid it elsewise when starting to program experiments

Questions?