Group Matching

Remixed from material by Ali Seyhun Saral & Philipp Chapkovski

Groups

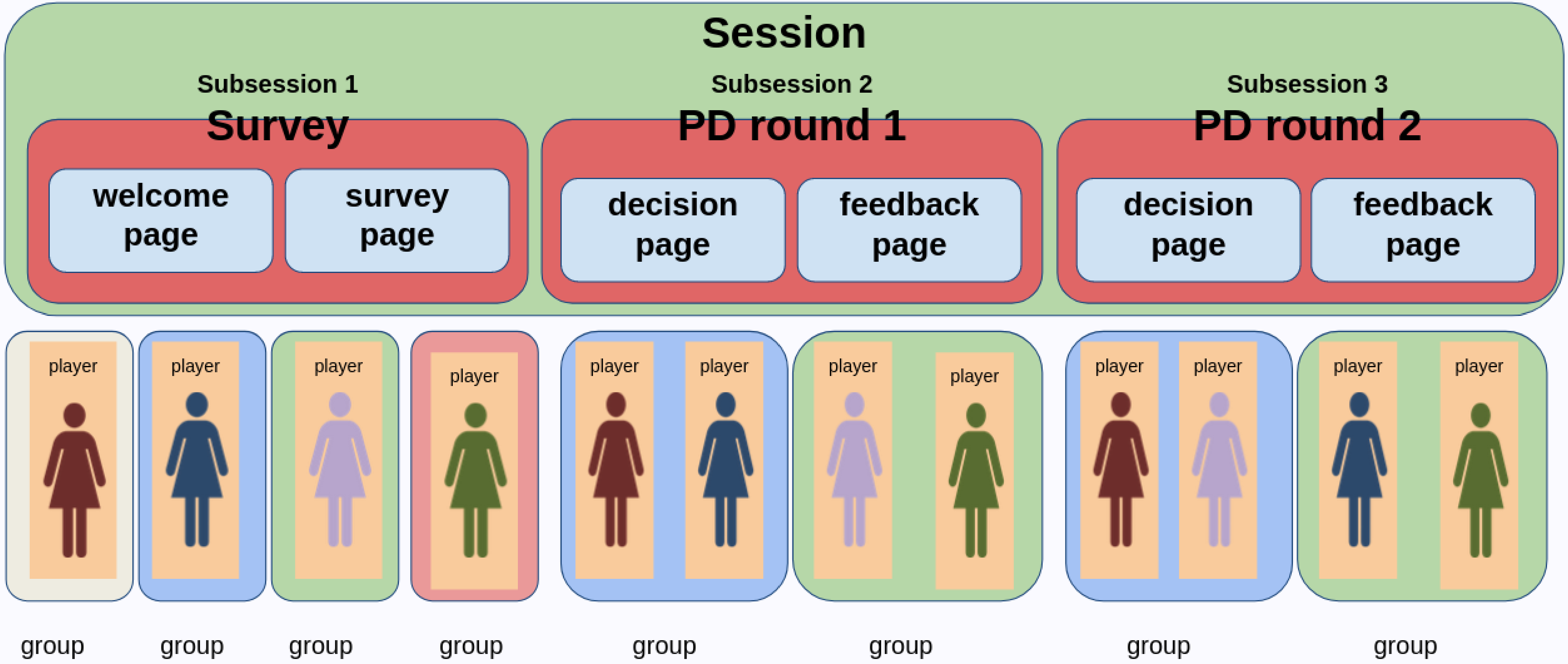
Last Lecture:

- Homogenous groups
- Heterogenous groups

This Lecture:

- Changing the group matching

Groups can change within the app

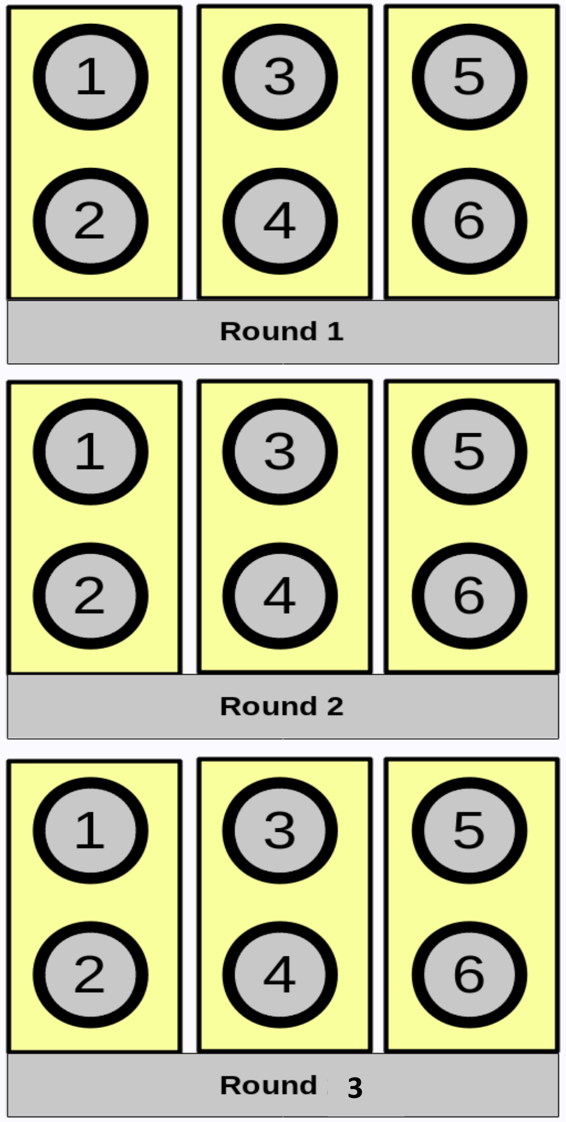


Default matching

```
class Constants(BaseConstants):
    name_in_url = 'some_name'
    players_per_group = 2
    num_rounds = 2
```

- oTree automatically creates groups based on the players_per_group variable
- Grouping sequentially by participant
- id_in_group also assigned sequentially

- Different ways to change grouping in the experiment
    - Usually done in the Subsession-method create_session()

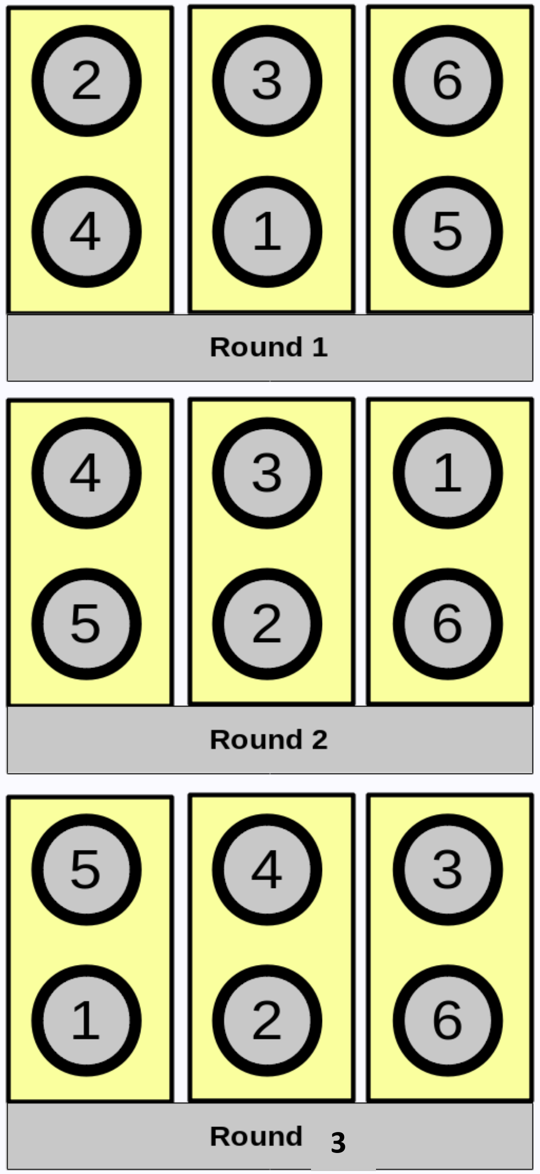


Random matching

```
class Subsession(BaseSubsession):

    def creating_session(self):
        self.group_randomly()
```

- group_randomly() shuffles players randomly for a given round
- Player can end up in any group, and **any position** within the group
    - Important to keep in mind when you have asymmetries or you use the position/id in the group for something else
- creating_session() is executed every round at the beginning of the experiment

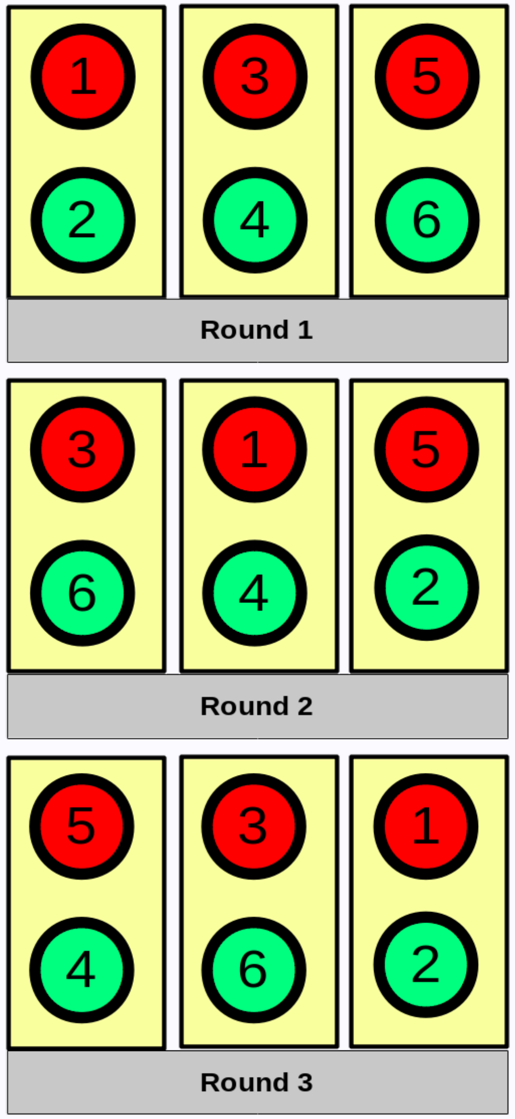


Random matching with fixed roles

```
class Subsession(BaseSubsession):

    def creating_session(self):
        self.group_randomly(fixed_id_in_group=True)
```

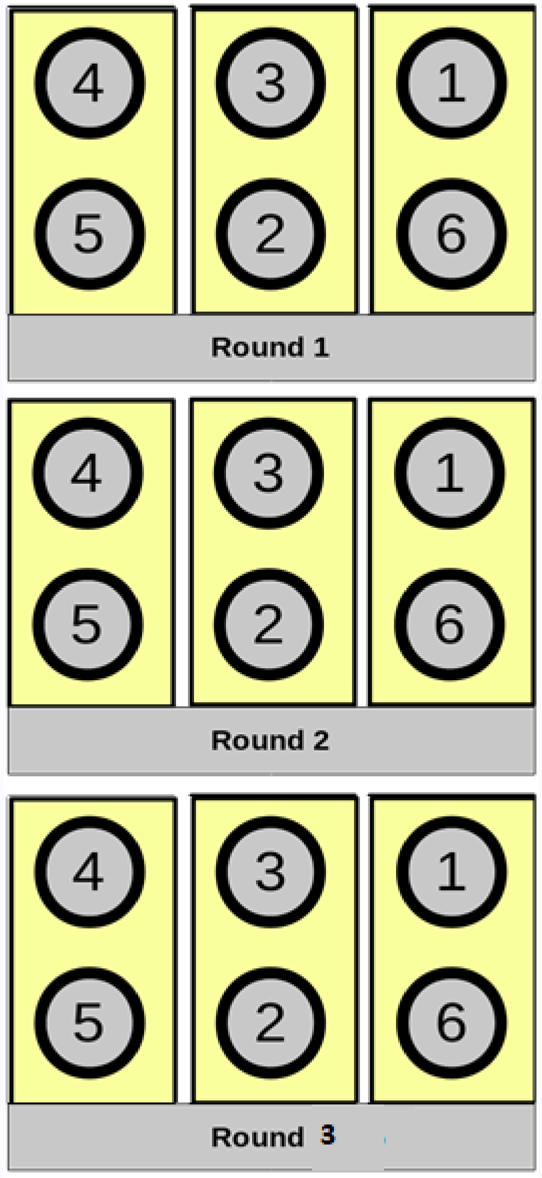
- Add fixed_id_in_group=True argument if you want to keep the roles of the player
  - Remember: Roles are assigned by id_in_group



Round-persistent random matching

```
class Subsession(BaseSubsession):
    def creating_session(self):
        if self.round_number == 1:
            self.group_randomly()
        else:
            self.group_like_round(1)
```

- group_like_round(n) creates a structure as in round n
- Possible use case:
  - Fixed matching within super game but random matching across super games



Specific matching

Two main methods for more complex matching:

<u>get_group_matrix():</u>

- return the structure of groups as a matrix
- list of lists, with each sublist being the player objects in a group, ordered by id_in_group

<u>set_group_matrix():</u>

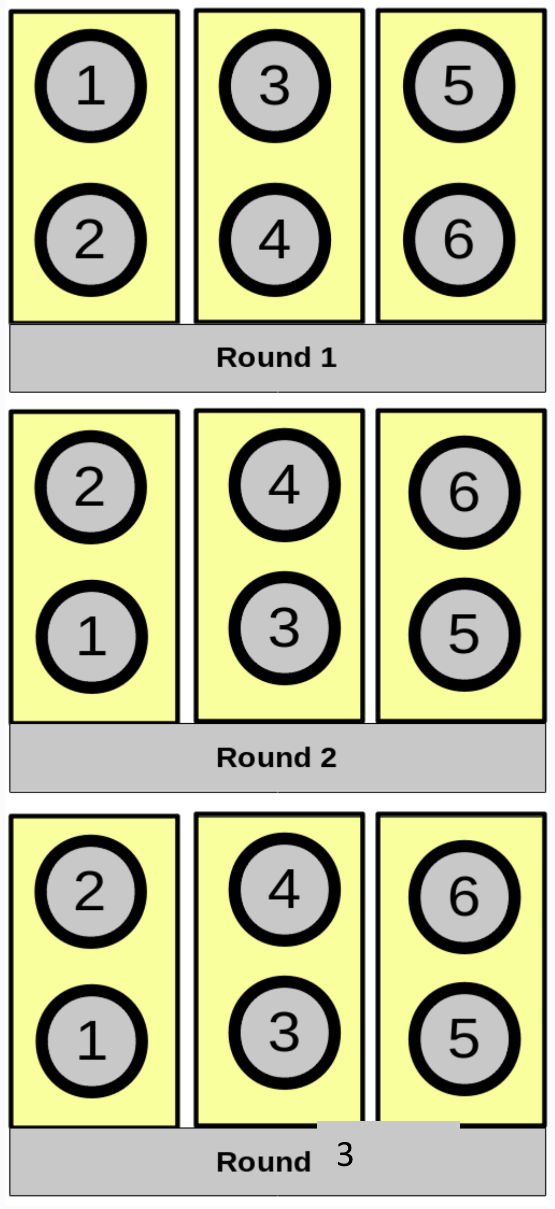
- lets you modify the group structure in any way you want
- Either get the group matrix and then manipulate it…
- … or providing a nested list of integers instead of player objects to create a matrix yourself
    - integer represents player's id_in_subsession

Specific matching example: Switch roles after first round

```
class Subsession(BaseSubsession):
   def creating_session(self):
       if self.round_number == 1:
           pass
       else:
           matrix = self.get_group_matrix()

          for row in matrix:
              row.reverse()

          self.set_group_matrix(matrix)
```

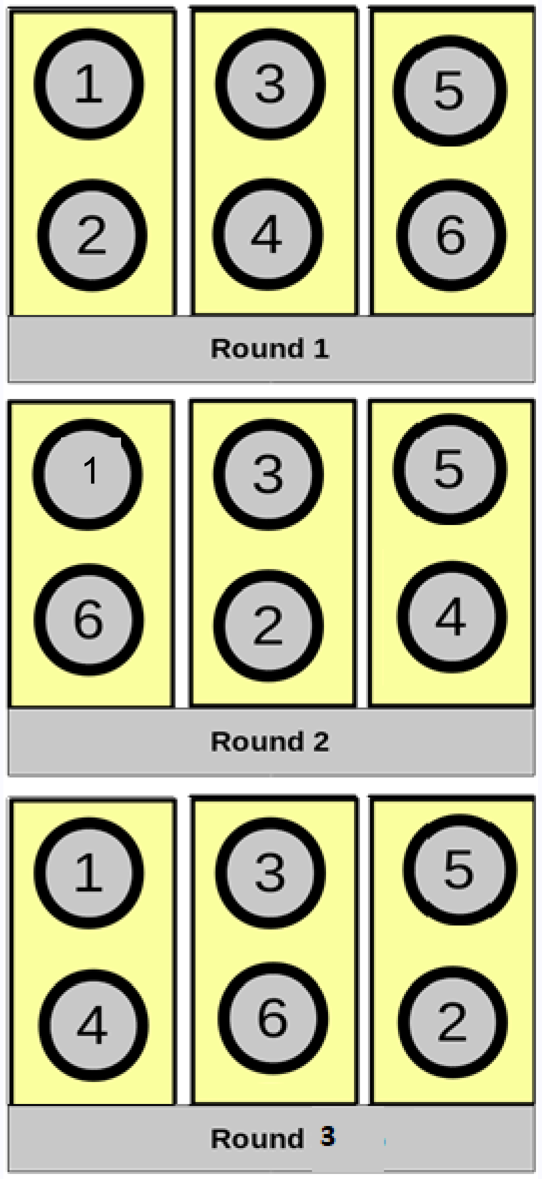


Specific matching example: Perfect stranger matching

```
class Subsession(BaseSubsession):
    def creating_session(self):
        structure_1 = [[1,2], [3,4], [5,6]]
        structure_2 = [[1,6], [3,2], [5,4]]
        structure_3 = [[1,4], [3,6], [5,2]]

       if self.round_number == 1:
            self.set_group_matrix(structure_1)
        elif self.round_number == 2:
            self.set_group_matrix(structure_2)
        elif self.round_number == 3:
            self.set_group_matrix(structure_3)
```

- set_group_matrix() can be used for any complex matching scheme you have in mind
- Either manipulate the matrix directly or pre-define lists which should correspond to id of the player in the subsession
- You can also use it to change the group size
- Note that creating_session() is executed **ONCE** when the session is created
    - Problem if grouping has to be conditional on choices of players/groups during the experiment



Shuffling during the session

```python
class ShuffleWaitPage(WaitPage):
    wait_for_all_groups = True
    after_all_players_arrive = 'do_my_shuffle'
```

- Define some method e.g. do_my_shuffle() in subsession that uses set_group_matrix
    - This method can use player choices etc
- Use it in the after_all_players_arrive method on a WaitPage to shuffle the players
- **Note that you have to wait_for_all_groups!!!**

Live Demo