Counterbalancing

2019/12/14

lecture 7

outline

1. big decisions
2. counterbalancing options
3. item effects

big decisions

- counterbalancing vs. randomization
- on the fly vs. predetermined

counterbalancing vs. randomization

counterbalancing

- deal with random effects by **systematically** arranging the uncontrollable
- e.g. assure that each stimulus is assigned to each condition an equal number of times across subjects
- especially helpful when dealing with **small numbers**
- more work (and room for errors)

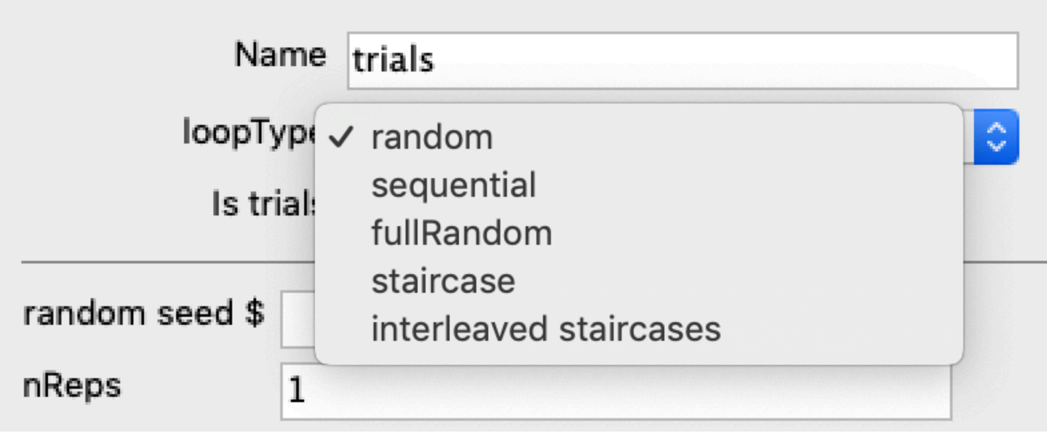
randomization

- deal with random effects by **randomly** assigning the uncontrollable
- e.g. randomly select stimuli to assign to conditions for each subject — shouldn't be biased on average
- can work well when working with **large numbers**
- less work (less room for errors)

counterbalancing vs. randomization

randomization

- Psychopy has built in randomization tools:
  - Builder lets you select trials random orders
  - Coder includes **TrialHandler** and **StairHandler** functions (http://www.psychopy.org/coder/codeTrials.html)
- Python's **random** package is also very helpful: https://docs.python.org/3/library/random.html
- Note that computers generate pseudo-random numbers, so **seeds** matter! When in doubt, seed with the current time to make each implementation unique



Name trials

loopType ✓ random

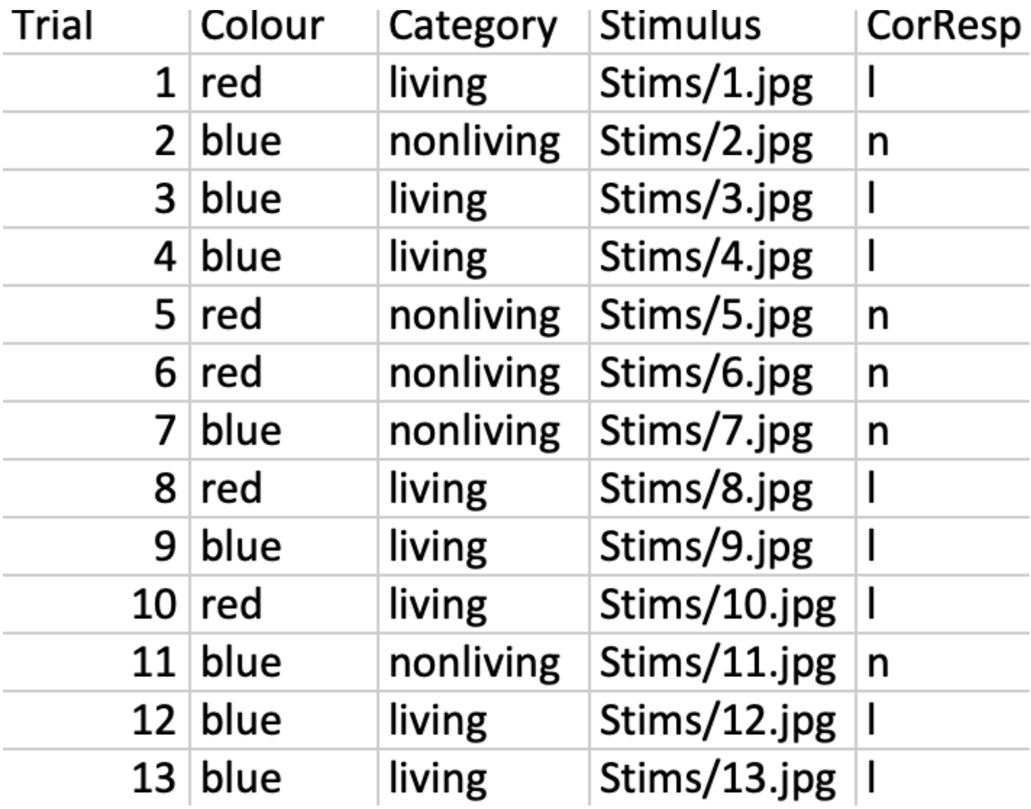
random seed $

nReps 1

on the fly vs. predetermined

predetermined

- write a script which generates experiment input files
- includes all counterbalancing/ randomization decisions
- each **row** details what exactly will happen on a **trial**
- can include stimulus, timing, condition, etc.
- often have many versions that you cycle over by subject
- **no surprises!**



Trial	Colour	Category	Stimulus	CorResp
1	red	living	Stims/1.jpg	l
2	blue	nonliving	Stims/2.jpg	n
3	blue	living	Stims/3.jpg	l
4	blue	living	Stims/4.jpg	l
5	red	nonliving	Stims/5.jpg	n
6	red	nonliving	Stims/6.jpg	n
7	blue	nonliving	Stims/7.jpg	n
8	red	living	Stims/8.jpg	l
9	blue	living	Stims/9.jpg	l
10	red	living	Stims/10.jpg	l
11	blue	nonliving	Stims/11.jpg	n
12	blue	living	Stims/12.jpg	l
13	blue	living	Stims/13.jpg	l

on the fly vs. predetermined

on the fly

- include functions/code in your **experiment script** that looks after all the counterbalancing/randomization while the experiment runs
- e.g. you want to counterbalance 24 trial-unique stimuli across three conditions:
    - use subject number counterbalancing condition: `cbNum = subnum%3`
    - rotate your stimulus list according to counterbalancing condition:

      ```
      rotation = cbNum * (24/3)
      cbStim = stim[-rotation:] + stim[:-rotation]
      ```
    - assign stimuli to conditions:

      ```
      cond1Stim = cbStim[0:8]
      cond2Stim = cbStim[8:16]
      cond3Stim = cbStim[16:24]
      ```

counterbalancing vs. randomization

predetermined

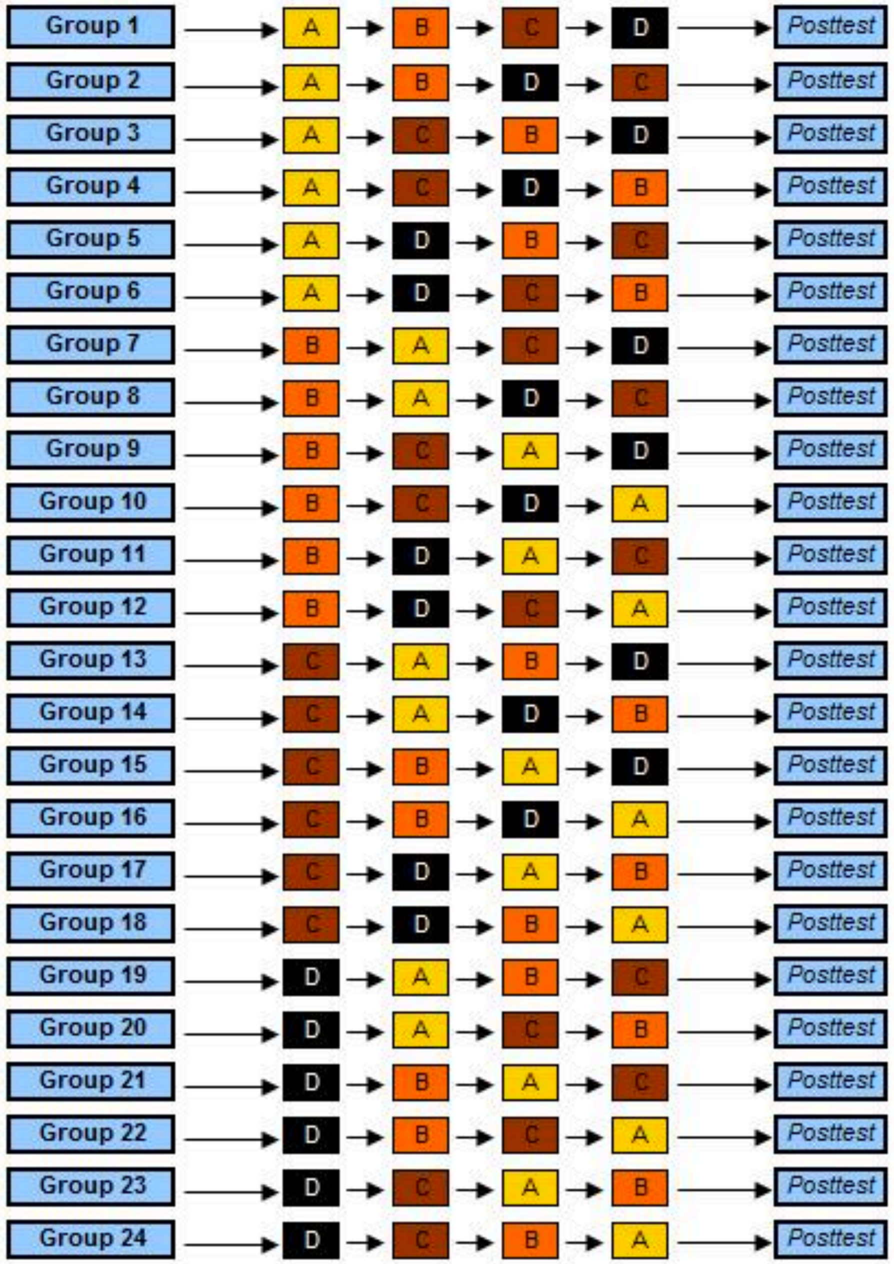
- can create “yoked” subjects
- you can combine all input files in advance to double check your counterbalancing/ randomization across the full sample
- feels more controlled

on the fly

- don’t have to worry about folders of auxiliary files
- can be used for more dynamic designs (e.g. stimuli based on prior decisions)
- code repository feels more integrated

counterbalancing options

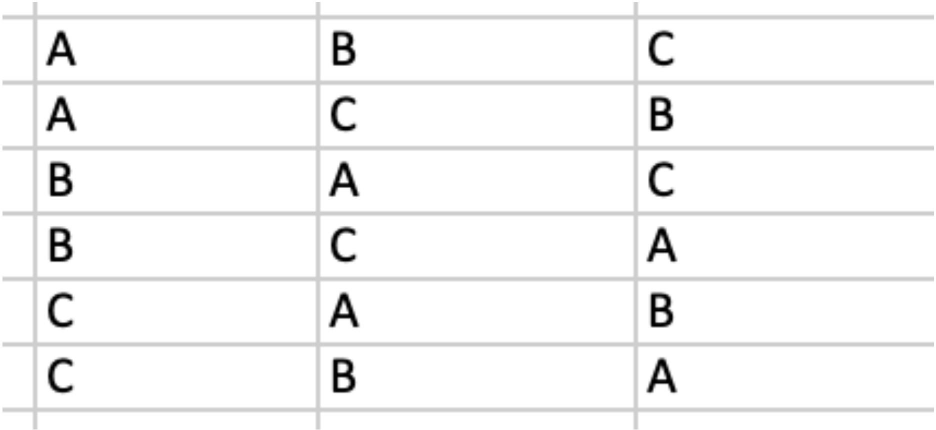
- complete counterbalancing
- incomplete (practical) counterbalancing



Group 1	A	B	C	D	Posttest
Group 2	A	B	D	C	Posttest
Group 3	A	C	B	D	Posttest
Group 4	A	C	D	B	Posttest
Group 5	A	D	B	C	Posttest
Group 6	A	D	C	B	Posttest
Group 7	B	A	C	D	Posttest
Group 8	B	A	D	C	Posttest
Group 9	B	C	A	D	Posttest
Group 10	B	C	D	A	Posttest
Group 11	B	D	A	C	Posttest
Group 12	B	D	C	A	Posttest
Group 13	C	A	B	D	Posttest
Group 14	C	A	D	B	Posttest
Group 15	C	B	A	D	Posttest
Group 16	C	B	D	A	Posttest
Group 17	C	D	A	B	Posttest
Group 18	C	D	B	A	Posttest
Group 19	D	A	B	C	Posttest
Group 20	D	A	C	B	Posttest
Group 21	D	B	A	C	Posttest
Group 22	D	B	C	A	Posttest
Group 23	D	C	A	B	Posttest
Group 24	D	C	B	A	Posttest

complete counterbalancing

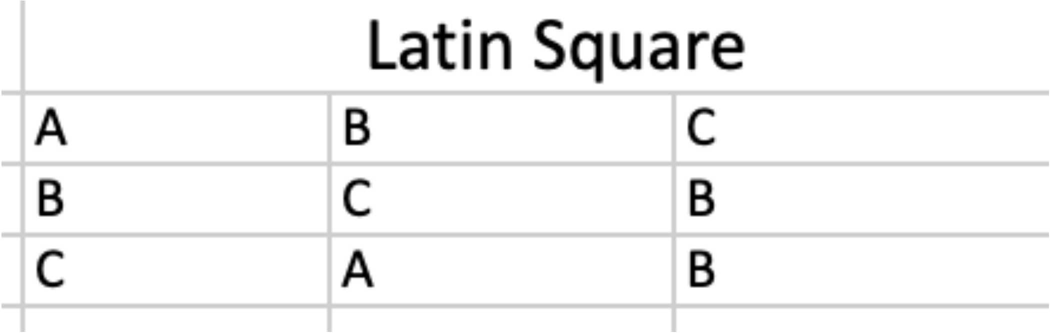
- **all possible sequences** of conditions are presented
- can be within or between participants
- deals with **position** effects:
  - people might get better with **practice**
  - people might **fatigue**
- deals with **order** effects:
  - condition B may be experienced different in **contrast** to A vs. C
  - processing from one condition may **carryover** to the next



A	B	C
A	C	B
B	A	C
B	C	A
C	A	B
C	B	A

incomplete (practical) counterbalancing

- only a **subset of possible sequences** are presented, often when total number of sequences gets too big
- **Latin Squares** efficiently deal with **position** effects
- **Complete Latin Squares** can deal with **position** and **order** effects by ensure that each condition occurs in each position once & after every other condition once.



Latin Square		
A	B	C
B	C	B
C	A	B

item effects

item effects

- **the stimuli assigned to conditions can matter!**
- some manipulations require that particular stimuli be assigned to particular conditions, e.g. natural categories
- if stimuli can be **repeated** across conditions, often optimal to ensure that each stimulus is presented in each condition an equal number of times **within subject**
- if stimuli need to be **trial-unique**, often need to ensure that they are equally likely to be presented in each condition **across subjects**