Problem Set 4

November 2016

1 More on the first experiment

In the first experiment, you ran several dot detection trials and stored the data collected from each trial in a matrix.

Let's have a quick recap about the experiment: each trial starts with a cross shape at the center of the screen, on which the subject must fix their eyes for 1s. After a randomly selected period of time, either a black dot or nothing will show up on the monitor for some amount of time between 50ms to 500ms. Finally, a green square will appear in the center of the screen to ask subjects to perform a decision.

Subjects are asked to press 'Left' immediately when they detect the dot, and 'Right' when they think no dot has been shown. Our goal for this part of the project is to measure the reaction time in this decision.

1.1 Code to measure reaction time

Below is a piece of code that measures reaction time when you are asked to press a button. You should pay attention to the variable RT in the code and understand how it works.

```
from psychopy import visual, event, core
from random import random

#create a window to draw in
wintype='pyglet' # use pyglet if possible, it's faster at event handling
myWin = visual.Window((600.0,600.0),allowGUI=False,winType=wintype)

# use the first font found on this list
sans = ['Gill Sans MT', 'Arial','Helvetica','Verdana']
ask4key=visual.TextStim(myWin, text='Press a key as fast as you can RIGHT NOW!', ...
    font=sans, units='norm', height=.10,pos=(-.95,0),alignHoriz='left')

RT = core.Clock() # make a clock for capturing RT (reaction time)

while True: # replace the ...s with spaces or a tab
```

```
# clear any keystrokes before starting
event.clearEvents()
allKeys=[]

# paint the stimulus to react to:
ask4key.draw()
myWin.flip()
RT.reset() # reaction time starting immediately after flip

while len(allKeys)==0: # wait for a keypress
    allKeys=event.getKeys(timeStamped=RT)
    # see the online PsychoPy reference manual on event.getKeys for details
    of use and code
    # if timeStamped = a core.Clock object, it causes return of the
    tuple (key,time-elapsed-since-last-reset)

# now allKeys is [(key, milliseconds)]
# if you don't have pyglet, you need to get the time explicitly
if not wintype == 'pyglet':
    allKeys[0][1] = RT.getTime()

# unpack allKeys  taking the first keypress in the list
thekey=allKeys[0][0].upper()
theRT =allKeys[0][1]

if thekey =='ESCAPE': core.quit()

if theRT < 0.5 : feedback="Pretty fast! Now wait for it..."
else: feedback="You can do better. C'mon now! ready....and ..."
msg="Your reaction time was %5.4f seconds. %s"%(theRT,feedback)
ask4key.setText(msg)
ask4key.draw()
myWin.flip()

ask4key.setText('Press a key as fast as you can RIGHT NOW!')
# wait for 3-8 seconds to read feedback and have uncertain
wait time before next trial

core.wait(3+int(random()*6))
```

1.2 Modify the code above

You may have noticed that the code we provided doesn't have a mechanism to prevent pressing the button without seeing the cue, which is something we would like to avoid. An easy way is to provide feedback when you detect this event, warning the subject not to do "cheat". Here we ask you to add this to the code.

Don't forget you should record the reaction times. You can borrow code from the previous problem set. This is essential since the next step is to plot and analyze the data.

Finally, you may also want to try something else, such as changing the cue. Feel free to explore yourselves!

1.3 Get back to the first experiment

Now we ask you to modify the code from problem set 3 so that you can measure the reaction time. Here are some hints:

```
RT.reset() # reaction time starting immediately after flip

    while len(k)==0: # wait for a keypress
        k=event.getKeys(timeStamped=RT)
        responseVector.append(k)
        core.wait(dotTime)
        win.flip()
        core.wait(1-dotTime)
        square = visual.Rect(win, width=50, height=50, fillColor=(0,255,0), ...
            fillColorSpace='rgb255')
        square.draw()
        win.flip()
```