Sample Psychopy Experiment

From Lupyan Lab

Here is a sample study in the format that we normally use in the Lupyan Lab. Most experiments depend on two scripts. The first generates the trial list. The second actually runs the study. You can download both scripts and the stimuli to run the experiment here (http://sapir.psych.wisc.edu/stimuli/sample-experiment.zip) .

Contents

Demo

Below is a demo of the experiment we are making. The video can be downloaded here (http://sapir.psych.wisc.edu/stimuli/sample-experiment-demo.mov) .

sample-exp

Generate Trial List

Helper Functions

For starters, we define some functions that will help us make the trial list.

```
#!/usr/bin/env python
"""
Makes a trial list given a subject code and a seed for the random generatc
"""

import random

def randomButNot(arr, index):
    """
    Function to select a random item of the array not equal to arr[index]
    """
        randIndex=index
        while randIndex == index:
                randIndex = random.randint(0, len(arr)-1)
        return arr[randIndex]

def circularList(lst,seed):
    """
    Makes an endless list.
    """
        if not isinstance(lst,list):
```

```
                lst = range(lst)
        i = 0
        random.seed(seed)
        while True:
                yield lst[i]
                if (i+1) % len(lst) ==0:
                        random.shuffle(lst)
                i = (i + 1)%len(lst)
```

Define Experiment-Specific Variables

Each generateTrial script is specific for each experiment. This script generates a trial list for a simple sound picture verification task in which participants hear a sound file through the headphones and then see a picture on the screen. Their job is to answer yes or no: does the sound match the picture.

```
# define categories of pictures
pics = ['baby','bee','bird','cat', 'chainsaw', 'dog','keyboard','scissors'

# identify categories as animate v. inanimate
isAnimate = ['1','1','1','1','0','1','0','0'] # note len(isAnimate) == len

# sound cues are the same categories as pics
cues = list(pics)

# Possible locations for the sound and the pictures
soundPicLocations = [
                                        ('left','left'),
                                        ('right','right'),
                                        ('left','right'),
                                        ('right','left')
                                        ]

#that's pretty tiresome to type. We don't need to permute by hand. Compute
#soundPicLocations = list(itertools.product(['left','right'],['left','righ

primeTypes = ["label","sound"] # types of sounds
picTypes = ["Silent","Sound"]  # types of pictures
picNum = ['1','2']             # two pictures of each categories
numIter = 2                    # how many times to repeat each unique tria
```

Generate Unique Trial Types

The main function in the generateTrials script creates all the unique trial types. Any ratios (e.g., proportion of valid to invalid trials) need to be defined here.

```
separator = ","
def     main(subjCode,seed):
    """
    Makes, shuffles, and writes a trial list to file.

    Subject code is used to write the file.
    Seed is used for pseudorandomization.
    """

    testFile = open('trialList_test_'+subjCode+ '.csv','w')

    # print column headers to the file
    print >> testFile, separator.join(("picType","primeType","picCategory"
        "picFile","soundCategory","soundFile","soundLocation","picLocation
        "isLocationCongruent","isMatch"))

    random.seed(seed)
    seed = int(seed)
    isMatch = circularList([1,1,1,0],seed) # ratio of match to mismatch is
    trialList =[]
    for curIter in range(numIter):
        for curSoundPicLocation in soundPicLocations:
            for curPicType in picTypes:
                for curPrimeType in primeTypes:
                    for curPic in pics:
                        for curPicNum in picNum:
                            curIsMatch = isMatch.next()
                            # if it's a match trial, the sound and the pic
                            # are from the same category
                            if curIsMatch==1:
                                curSound = curPic
                                if curSoundPicLocation[0]=='center':
                                    curSoundLocation = ''
                                else:
                                    curSoundLocation = '_'+curSoundPicLoca
                                    curSoundFile = curPic+"_"+str(curPrime
                                    curPicFile = curPicType+"_"+curPic+cur

                            # if it's a mismatch trial, the sound and the
                            # are from different categories
                            elif curIsMatch==0:
                                curSound = randomButNot(pics, pics.index(c
                                curSoundFile = curSound+"_"+str(curPrimeTy
                                curPicFile = curPicType+"_"+curPic+curPicN

                            # make sure to append the variables to the tri
                            trialList.append(separator.join((curPicType, c
                                curPicFile, curSound,curSoundFile, curSoun
                                str(int(curSoundPicLocation[0]==curSoundPi
```

```
    # shuffle all the trials and print
    # seeding the random number generator ensures we can get the same ranc
    random.shuffle(trialList)
        for curTrialList in trialList:
                print >>testFile, curTrialList

if __name__ == "__main__":
    """
    Running the file as a script (instead of importing it) creates a sampl
    """
    trialList = main('testTrials-15',15)
```

Main experiment script

The main experiment uses Psychopy for stimulus presentation and response recording. The experiment setup happens in two parts: getting the session variables, and loading all the materials for the experiment. The bulk of the experiment is simply looping through the trial list one row/trial at a time.

Get Session Variables

```
#!/usr/bin/env python
import time
from baseDefsPsychoPy import * # custom LupyanLab functions
from stimPresPsychoPy import *  # custom LupyanLab functions
import generateTrials

class Exp:
    def __init__(self):

        #this is where the subject variables go.  'any' means any value is
        # allowed as long as it's the correct type (str, int, etc.) the nu
        # 1 and 2 control the order in which the prompts are displayed (di
        # have no natural order)

        self.optionList = { '1':  {   'name' : 'subjCode',
                                    'prompt' : 'Subject Code: ',
                                   'options': 'any',
                                  'default' : 'systemA_101',
                                     'type' : str},
                            '2' : {   'name' : 'gender',
                                    'prompt' : 'Subject Gender m/f: ',
                                   'options' : ("m","f"),
                                    'default':'',
                                     'type' : str},
                            '3' : {   'name' : 'responseDevice',
                                    'prompt' : 'Response device: keyboard/
                                   'options' : ("keyboard","gamepad"),
                                   'default' : 'gamepad',
```

```
                                    'type' : str},
                            '4' : {   'name' : 'whichValid',
                                    'prompt' : 'whichKeyValid "up" or "dow
                                   'options' : ('up','down'),
                                    'default':'',
                                      'type' : str},
                            '5' : {   'name' : 'useFeedback',
                                    'prompt' : 'Use Feedback? y/n ',
                                   'options' : ("y","n"),
                                   'default' : 'y',
                                      'type' : str},
                            '6' : {   'name' : 'seed',
                                    'prompt' : 'Enter seed: ',
                                   'options' : 'any',
                                   'default' : 100,
                                      'type' : int},
                            '8' : {   'name' : 'expInitials',
                                    'prompt' : 'Experiment Initials: ',
                                   'options' : 'any',
                                   'default' : '',
                                      'type' : str}
                        }

        optionsReceived=False
        fileOpened=False
        while not optionsReceived or not fileOpened:
            # enterSubjInfo is a Lupyan-Lab specific wrapper for a psychop
            [optionsReceived,self.subjVariables] = enterSubjInfo('picWordV
            if not optionsReceived:
                popupError(self.subjVariables)
            try:
                if  os.path.isfile(self.subjVariables['subjCode']+'_test.t
                    fileOpened=False
                    popupError('Error: That subject code already exists')
                else:
                    self.outputFileTest     = file(self.subjVariables['sub
                    fileOpened=True
            except:
                pass

        # generate the trials, which writes out to a .csv
        generateTrials.main(self.subjVariables['subjCode'],self.subjVariab

        if self.subjVariables['responseDevice']=='gamepad':
            try:
                self.stick=initGamepad()
                pygame.init()
                self.inputDevice = "gamepad"
                responseInfo = " Press the Green button for 'Yes' and the
                self.validResponses = {'0':3,'1':0}
            except SystemExit:
```

```
                self.subjVariables['responseDevice']='keyboard'
                print "No joystick; using keyboard"
                self.inputDevice = "keyboard"
                if self.subjVariables['whichValid']=='up':
                    self.validResponses = {'1':'up','0':'down'} #change n/
                    responseInfo = " Press the 'up arrow' for 'Yes' and th
                else:
                    self.validResponses = {'0':'up','1':'down'} #change n/
                    responseInfo = " Press the 'down arrow' for 'Yes' and
        else:
            print "Using keyboard"
            self.inputDevice = "keyboard"
            if self.subjVariables['whichValid']=='up':
                self.validResponses = {'1':'up','0':'down'} #change n/o tc
                responseInfo = " Press the 'up arrow' for 'Yes' and the 'd
            else:
                self.validResponses = {'0':'up','1':'down'} #change n/o tc
                responseInfo = " Press the 'down arrow' for 'Yes' and the

        try:
            self.win = visual.Window(fullscr=True, color=[.6,.6,.6], allow
        except:
            self.win = visual.Window([1024,768], color=[.6,.6,.6], allowGU

        self.preFixationDelay   =     0.750
        self.postFixationDelay  =     0.500

        self.stimPositions = {'center':(0,0), 'left':(-570,0),'right':(570
        self.numPracticeTrials = 4
        self.takeBreakEveryXTrials = 100;
        self.finalText              = "You've come to the end of the exper
        self.instructions        = \
        """Thank you for participating \nIn this experiment you will hear
        a meowing sound and see a cat. Other times, the picture will not m
        During the experiment there will be some breaks during which you c
        """
        self.instructions+=responseInfo

        self.takeBreak = "Please take a short break.  Press 'Enter' when y
        self.practiceTrials = "The next part is practice"
        self.realTrials = "Now for the real trials"
```

Load Stimuli

After we get the subject info and make the trials file, we can initialize the experiment by loading all the stimuli into Psychopy objects.

```
class trial(Exp):
        def __init__(self):
```

```
                firstStim=''

class ExpPresentation(trial):
        def __init__(self,experiment):
                self.experiment = experiment

        def initializeExperiment(self):
                # Experiment Clocks
                self.expTimer = core.Clock()
                """This loads all the stimili and initializes the trial se
                self.fixSpot = visual.TextStim(self.experiment.win,text="+
                self.centerRectOuter = newRect(self.experiment.win,size=(2
                self.centerRectInner = newRect(self.experiment.win,size=(2
                showText(self.experiment.win, "Loading Images...",color="b
                self.pictureMatrix = loadFiles('stimuli\Pictures','jpg','i
                self.soundMatrix = loadFiles('stimuli\Sounds','wav','winSc
                (self.trialListMatrix,self.fieldNames) = importTrials('tri
                self.stim = visual.PatchStim(self.experiment.win,mask="non
        def checkExit(self): #I don't think this works if gamepad is in us
                if event.getKeys()==['equal','equal']:
                        sys.exit("Exiting experiment")
```

Trial Procedure

Behavior on each trial is defined by a single function called "presentTestTrial". It's main argument is a dict-like object that corresponds to a row from the trial list.

```
def presentTestTrial(self,whichPart,curTrial,curTrialIndex):

    self.checkExit() #check for exit press the equals key twice.
    self.experiment.win.flip()
    core.wait(self.experiment.preFixationDelay)
    #setAndpresentStimulus(self.experiment.win,[self.fixSpot]) #show fixat
    core.wait(self.experiment.postFixationDelay)

    playAndWait(self.soundMatrix[curTrial['soundFile']],soundPath=self.sou
    core.wait(.8)
    print 'soundLocation is',curTrial['soundLocation'],'picLocation is',cu

    self.pictureMatrix[curTrial['picFile']][0].setPos(self.experiment.stim
    setAndPresentStimulus(self.experiment.win,[self.pictureMatrix[curTrial

    correctResp = self.experiment.validResponses[str(curTrial['isMatch'])]
    if self.experiment.inputDevice=='keyboard':
        (response,rt) = getKeyboardResponse(self.experiment.validResponses
    elif self.experiment.inputDevice=='gamepad':
        (response,rt) = getGamepadResponse(self.experiment.stick,self.expe

    #self.soundMatrix[curTrial['soundFile']].stop()
```

```
    if response==correctResp:
        isRight=1
        if self.experiment.subjVariables['useFeedback']=='y':
            playAndWait(self.soundMatrix['bleep'],winSound=True)
    else:
        isRight=0
        if self.experiment.subjVariables['useFeedback']=='y':
            playAndWait(self.soundMatrix['buzz'],winSound=True)

    self.experiment.win.flip()
    fieldVars=[]
    for curField in self.fieldNames:
        fieldVars.append(curTrial[curField])
    curLine = createResp(self.experiment.optionList,self.experiment.subjVa
    a_whichPart = whichPart,
    b_curTrialIndex = curTrialIndex,
    c_expTimer = self.expTimer.getTime(),
    d_isRight = isRight,
    e_rt = rt*1000)
    writeToFile(self.experiment.outputFileTest,curLine)
```

Cycle Through Experiment

The rest of the experiment is simply looping through the rows of the trial list and presenting each trial. Here we cycle through the experiment and behave differently if it is a practice trial or a test trial.

```
def cycleThroughExperimentTrials(self,whichPart):
    if whichPart == "practice":
        trialIndices = random.sample(range(1,50),self.experiment.numPracti
        curTrialIndex=0
        for curPracticeTrial in trialIndices:
            self.presentTestTrial(whichPart,self.trialListMatrix.getFuture
    else:
        curTrialIndex=0
        for curTrial in self.trialListMatrix:
            self.checkExit()
            if curTrialIndex>0 and curTrialIndex % self.experiment.takeBre
                showText(self.experiment.win,self.experiment.takeBreak,col
            self.presentTestTrial(whichPart,curTrial,curTrialIndex)
            curTrialIndex+=1
        #close test file
        self.experiment.outputFileTest.close()
```

Run Experiment

The bottom of the script outlines the order of events for the experiment. When the experiment object is created, a GUI pops up to get the session info. Then the trials are made, stimuli are loaded, and the devices are created. The instructions are shown using a Lupyan Lab specific function called "showText" which is a wrapper for making an on-the-fly psychopy.visual.TextStim object and collecting input from either the keyboard, mouse, or gamepad. Then the practice trials are run, followed by the test trials.

```
currentExp = Exp()
currentPresentation = ExpPresentation(currentExp)
currentPresentation.initializeExperiment()
showText(currentExp.win,currentExp.instructions,color=(-1,-1,-1),inputDevi
showText(currentExp.win,currentExp.practiceTrials,color=(-1,-1,-1),inputDe
currentPresentation.cycleThroughExperimentTrials("practice")
showText(currentExp.win,currentExp.realTrials,color=(0,0,0),inputDevice=cu
currentPresentation.cycleThroughExperimentTrials("test")
showText(currentExp.win,currentExp.finalText,color=(-1,-1,-1),inputDevice=
```