Reaction times

christophe@pallier.org

Sept. 2013

Contents

(1) Write a program, using pygame, that starts with an empty black screen and after one second, displays a small white dot at the center of the screen for 1 second, waits for another second and then quits (tip: use the function pygame.time.delay to wait for a given amount of time)

```
#! /usr/bin/env python
# -*- coding:utf-8 -*-
# Time-stamp: <2013-09-14 21:13 christophe@pallier.org>


# initialisation de pygame
import pygame
pygame.init()

fullscreen = False
if fullscreen:
    screen = pygame.display.set_mode((0,0),pygame.FULLSCREEN|pygame.DOUBLEBUF)
else:
    screen = pygame.display.set_mode((600,400),pygame.DOUBLEBUF)

r = screen.get_rect()
W, H = r.width, r.height
pygame.mouse.set_visible(False)

###################################

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

screen.fill(BLACK)
```

```
pygame.display.update()

pygame.time.delay(1000)

pygame.draw.circle(screen, WHITE, (W/2, H/2), 4)
pygame.display.update()

pygame.time.wait(1000)


screen.fill(BLACK)
pygame.display.update()

pygame.time.wait(1000)

pygame.quit()
```

---

(2) Start the following program, move the mouse into the pygame window and clik on it several times, press keys, ... close the window to quit.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((640, 400))

while True:
    event = pygame.event.wait()
    if event.type == pygame.QUIT:
        break
    print(pygame.event.event_name(event.type))

pygame.quit()
```

(you can skim through the tutorial at http://lorenzod8n.wordpress.com/2007/05/30/pygame-tutorial-3-mouse-events/) to understand pygame's mouse events. Now, modify the previous code so that the dot remains on the screen until you press a mouse button.

```
#! /usr/bin/env python
# -*- coding:utf-8 -*-
# Time-stamp: <2013-09-16 18:53 christophe@pallier.org>

# initialisation de pygame
import pygame
pygame.init()

fullscreen = False
if fullscreen:
    screen = pygame.display.set_mode((0,0),pygame.FULLSCREEN|pygame.DOUBLEBUF)
else:
    screen = pygame.display.set_mode((600,400),pygame.DOUBLEBUF)

r = screen.get_rect()
W, H = r.width, r.height
pygame.mouse.set_visible(False)

##################################

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

screen.fill(BLACK)
pygame.display.update()

pygame.time.delay(1000)

pygame.draw.circle(screen, WHITE, (W/2, H/2), 4)
pygame.display.update()

pygame.event.clear()
while True:
    ev = pygame.event.poll()
    if ev.type == pygame.MOUSEBUTTONDOWN:
        break

screen.fill(BLACK)
pygame.display.update()

pygame.time.wait(1000)

pygame.quit()
```

(3) Look at the documentation of the function pygame.time.get_ticks() and add code to measure and print the reaction-times, that is the time between the onset of the dot's display and the mouse button press.

```
#! /usr/bin/env python
# -*- coding:utf-8 -*-
# Time-stamp: <2013-09-16 18:54 christophe@pallier.org>

# initialisation de pygame
import pygame
pygame.init()

fullscreen = False
if fullscreen:
    screen = pygame.display.set_mode((0,0),pygame.FULLSCREEN|pygame.DOUBLEBUF)
else:
    screen = pygame.display.set_mode((600,400),pygame.DOUBLEBUF)

r = screen.get_rect()
W, H = r.width, r.height
pygame.mouse.set_visible(False)

##################################

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

screen.fill(BLACK)
pygame.display.update()

pygame.time.delay(1000)

pygame.draw.circle(screen, WHITE, (W/2, H/2), 4)
pygame.display.update()

t0 = pygame.time.get_ticks()
pygame.event.clear()
while True:
```

```
    ev = pygame.event.poll()
    if ev.type == pygame.MOUSEBUTTONDOWN:
        rt = pygame.time.get_ticks() - t0
        print(rt)
        break

screen.fill(BLACK)
pygame.display.update()

pygame.time.wait(1000)

pygame.quit()
```

---

(4) Modify the previous program to repeat the process for 32 trials, and:

- Make the time between the appearance of two dots a random duration between 1 and 2 secs (tip: use the function random.randint).
- Record the reaction times in a list.
- At the end of the program, save the reaction times in a text file.
- Run this program twice using your right hand and twice using your left hand, *trying to respond as fast as possible, but without making false alarms* (change the name of the text file to save into different text files).

```
#! /usr/bin/env python
# -*- coding:utf-8 -*-
# Time-stamp: <2013-09-17 18:41 christophe@pallier.org>


import random, sys

# initialisation de pygame
import pygame
pygame.init()

fullscreen = False
if fullscreen:
```

```
    screen = pygame.display.set_mode((0,0),pygame.FULLSCREEN|pygame.DOUBLEBUF)
else:
    screen = pygame.display.set_mode((600,400),pygame.DOUBLEBUF)

r = screen.get_rect()
W, H = r.width, r.height
pygame.mouse.set_visible(False)

###################################

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

screen.fill(BLACK)
pygame.display.update()

reaction_times = []
NTRIALS = 32

for trial in range(NTRIALS):
    waitingtime = random.randint(1000, 2000)
    pygame.time.delay(waitingtime)

    pygame.draw.circle(screen, WHITE, (W/2, H/2), 4)
    pygame.display.update()

    t0 = pygame.time.get_ticks()
    pygame.event.clear()
    while True:
        ev = pygame.event.poll()
        if ev.type == pygame.QUIT:
            sys.exit()
        if ev.type == pygame.MOUSEBUTTONDOWN:
            rt = pygame.time.get_ticks() - t0
            print(rt)
            break

    screen.fill(BLACK)
    pygame.display.update()

    reaction_times.append(rt)

# sauvegarde des temps de reaction
rtfile = file("reaction-times_l1.csv", "w")
for r in reaction_times:
    rtfile.write(str(r) + chr(10)) # chr(10) is a newline
```

```
rtfile.close()

pygame.quit()
```

---

(5) Look at your reaction time data. If you know to do it, plot histograms with a spreadsheet program or R (or any other software)

Launch spyder. Go to the console window, in the options, set the working directory to the directory that contains reaction-times.csv. Type the following commands:

```
f = open('reaction-times.csv')
l = []
for line in f:
    l.append(int(line))
l
plot(l)
hist(l)
mean(l)
median(l)
```

---

(6) Back to the reaction time program, write a function that removes the 25% extreme data points in a list and returns the mean and standard errors of the remaining data points (not using numpy).

```
#! /usr/bin/env python
# -*- coding:utf-8 -*-
# Time-stamp: <2013-09-16 21:05 christophe@pallier.org>


import random, sys

# initialisation de pygame
```

```
import pygame
pygame.init()

fullscreen = False
if fullscreen:
    screen = pygame.display.set_mode((0,0),pygame.FULLSCREEN|pygame.DOUBLEBUF)
else:
    screen = pygame.display.set_mode((600,400),pygame.DOUBLEBUF)

r = screen.get_rect()
W, H = r.width, r.height
pygame.mouse.set_visible(False)

##################################

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

screen.fill(BLACK)
pygame.display.update()

reaction_times = []
NTRIALS = 32

for trial in range(NTRIALS):
    waitingtime = random.randint(1000, 2000)
    pygame.time.delay(waitingtime)

    pygame.draw.circle(screen, WHITE, (W/2, H/2), 4)
    pygame.display.update()

    t0 = pygame.time.get_ticks()
    pygame.event.clear()
    while True:
        ev = pygame.event.poll()
        if ev.type == pygame.QUIT:
            sys.exit()
        if ev.type == pygame.MOUSEBUTTONDOWN:
            rt = pygame.time.get_ticks() - t0
            print(rt)
            break

    screen.fill(BLACK)
    pygame.display.update()

    reaction_times.append(rt)
```

```
# sauvegarde des temps de reaction
rtfile = file("reaction-times.csv", "w")
for r in reaction_times:
    rtfile.write(str(r)+"n")
rtfile.close()

def mean(l):
    sum = 0.0
    for x in l:
        sum = sum + x
    return sum/len(l)

import math

def stderr(l):
    m = mean(l)
    n = len(l)
    sumsq = 0.0
    for x in l:
        sumsq = sumsq + (x - m)**2
    return math.sqrt(sumsq/n)/math.sqrt(n)

def tmean(l, percentage=0.25):
    sl = sorted(l)
    trimmed = int(len(l) * percentage/2.0)
    clean = []
    for i in range(trimmed, len(l)-1-trimmed):
        clean.append(sl[i])
    return (mean(clean), stderr(clean))


print("Trimmed mean = " + str(tmean(reaction_times)))

pygame.quit()
```

---

(7) Change the size of the dot so that it is 10 times bigger than previously.

Rerun the experiment. How is your reaction times affected?

(8) Now, change the contrast of the dot with the background (make it grey rather than white). How is your reaction times affected?

(9) Modify the previous program so that the color of target is randomly chosen between blue and red (with equal probability) at each trial. You still use only one response button.

(10) The task is to now to take a decision: one mouse button is assigned to blue and another one to red (if you only have one mouse button, use 2 keys and the KEYDOWN event). In the results file, save both the button pressed and the reaction times (on row per trial). Compare decision times to simple reaction times.

(11) Modify the program so that the dot can appear (randomly) at one of two locations far on the left or far on the right of the center. How is your reaction time affected?

(12) (advanced): Modify the simple detection program so that the contrast can be modified at each trial. Write the program to find the threshold of detection of the do.