Assignment 2

NESC 3505

Fall 2021-22

Instructions

This assignment is a sequence of questions (in Markdown cells) and cells for you to enter your answers. Each code cell where you should put an answer shows as:

```
    # YOUR CODE HERE
    raise NotImplementedError()
```

Be sure to *replace* the contents of those cells with your answer. Don't leave those lines of code in or you'll get errors. However, be careful when you're removing that code, to *only* remove what's shown above

Submitting your assignment

You can work on your assignment, on CoCalc, up until the due date and time, at which point we will collect your work. So you don't have to do anything on CoCalc to submit the assignment. The assignment also shows on Teams, but again you don't need to submit anything on Teams - that just allows us to track all your grades in one place. We will post your grade to Teams.

Late work

As stated in the syllabus, there is a late penalty of 2%/hour. Since we automatically collect your work at the due date/time, if you wish for late work to be considered, you'll need to contact the instructor (via Teams) *prior to the due date/time* to request an extension. Extensions are granted at the discretion of the instructor.

Reaction Time Data

The cell below contains reaction times (in seconds) from some trials in a behavioural experiment. Execute the cell (shift-enter) and then read the instructions in the cell below it.

In [63]:
```
# This cell contains the setup you need to complete the assignment.
# Run this cell before moving on

rt = [0.394252808, 0.442094359, 0.534764366, 0.565906723, 0.570404592,
      0.486154719, 0.518792127, 0.844916827, 0.495622859, 0.476159436,
      0.612854746, 0.529661203, 0.389157455, 1.517088266, 0.573962432,
      0.714152493, 0.409225638, 0.435308188, 0.509801957, 0.544626271,
      0.437877745, 0.333356848, 0.401773569, 0.479840688]
```

Question 1

What *type* of data is `rt` (in terms of Python data types)? Use a Python command to generate the answer.

In [4]:
```
#List
print(type(rt))
```

Out[4]:
```
<class 'list'>
```

Question 2

What *type* of data is the first value in `rt` (in terms of Python data types)? Use a Python command to generate the answer.

In [10]:
```
#Float
print(type(rt[0]))
```

Out[10]:
```
<class 'float'>
```

Question 3

How many trials were in this experiment? (Hint: how many entries are there in `rt`?). Use Python code to generate the answer.

In [11]:
```
#24
print(len(rt))
```

Out[11]:
```
24
```

Question 4

Print the first 9 values in `rt`

In [4]:
```
# 0.394252808, 0.442094359, 0.534764366, 0.565906723, 0.570404592, 0.486154719,
0.518792127, 0.844916827, 0.495622859
print(rt[0:9])
```

Out[4]:
```
[0.394252808, 0.442094359, 0.534764366, 0.565906723, 0.570404592, 0.486154719,
0.518792127, 0.844916827, 0.495622859]
```

Question 5

Print the last 6 values in `rt`

In [6]:
```
# 0.509801957, 0.544626271, 0.437877745, 0.333356848, 0.401773569, 0.479840688
print(rt[-6:])
```

Out[6]:
```
[0.509801957, 0.544626271, 0.437877745, 0.333356848, 0.401773569, 0.479840688]
```

Question 6

Print the values of the fifteenth through twentieth data points in `rt` (including the twentieth value)

In [7]:
```
#0.573962432, 0.714152493, 0.409225638, 0.435308188, 0.509801957, 0.544626271
print(rt[14:20])
```

Out[7]:
```
[0.573962432, 0.714152493, 0.409225638, 0.435308188, 0.509801957, 0.544626271]
```

Question 7

What is the *slowest* reaction time in `rt`?

In [8]:
```
# 0.333356848
print(min(rt))
```

Out[8]:
```
0.333356848
```

Question 8

What is the *fastest* reaction time in `rt`?

In [9]:
```
# 1.517088266
print(max(rt))
```

Out[9]:
```
1.517088266
```

Question 9

You cand find the index of a specific value in a list using the `.index()` method. Do this to find which data point (index) in `rt` has the value of 0.409225638

In [2]:
```
# 16
rt.index(0.409225638)
```

Out[2]:
```
16
```

Accuracy Data

In behavioural experiments it's common to analyze both reaction times and error rates. The list below contains a value for each trial indicating whther the subject made an error (`True`) or not (`False`).

Note that it might be more intuitive if this were coded as `True` for correct responses, and `False` for errors, but the variable is recording errors, not accuracy. It's always important in data science to make sure you undersatnd what your data represent!

In [64]:
```
# Just run this cell; don't change anything in it

err = [False, False, True, False, False, False, False, False, True, False,
       False, True, False, False, False, False, True, True, True, False,
       ]
```

Question 10

What Python data type are the *values* of `err`? (not the type of `err` itself). Use code to generate your answer, showing the type of the first entry in the `err` list.

In [6]:
```
#Boolean
print(type(err[0]))
```

Out[6]:
```
<class 'bool'>
```

Question 11

How many data points do we have in `err`?

In [7]:
```
#20
print(len(err))
```

Out[7]:
```
20
```

Question 12

These data are from the same experiment/participant as the RT data, but you'll note we have fewer data points in `err`. Let's say this is because of some sort of technical error during data recording, but we know (never mind how - this is just for the assignment!) what the missing data should be. Specifically, the first data point is missing, but we

know the participant made an error on that trial; and the last three data points are missing, and we know the participant got all of those trials correct.

Write four lines of code, as follows:

1. Insert a value at the beginning of `err` (without changing any of the existing values) to reflect the participant's error on the first trial.
2. Insert three values (using one line of code) at the end of `err`, indicating correct answers on the last three trials.
3. Print out `err` with these changes made.
4. Print out the length of `err` to confirm that it now matches the length of `rt`

Warning: Be aware that if you try this and it doesn't work the first time, or if you re-run the cell for any reason, your length may not match the length of RT. You can "reset" the values/length of `err` by re-running the cell above where `err` was first assigned.

In [65]:
```
err.insert(0, True)
err.extend([False,False,False])
print(err)
print('err=',len(err))
print('rt=',len(rt))
```

Out[65]:
```
[True, False, False, True, False, False, False, False, False, True, False, False,
True, False, False, False, False, True, True, True, False, False, False, False]
err= 24
rt= 24
```

Question 13

How many errors did the participant make? Use code — and specifically a list method — to generate the answer. This method was not covered in the lesson, so you may need to use the `help` command to figure out which method is approrpiate.

In [50]:
```
# YOUR CODE HERE
print('Errors=',err.count(True))
```

Out[50]:
```
Errors= 7
```

Data Cleaning

It is not uncommon in behavioural studies (or other research) to have *outliers* — one or a few values that are exceptionally different from the majority of values. These can be problematic for statistical analysis, and may also not reflect the behaviour we're trying to measure. For example, an RT may be exceptionally long because the participant sneezed prior to pressing the button.

Above, you should have identified the longest RT in this data, which is almost 1 s longer than any other RT. We would like to remove it from the data. When we do this, we should also remove the corresponding trial's data from the error data.

Question 14

Write code that does the following:

- finds the position (index) of the slowest RT in the data

- removes that slowest RT value from `rt`
- removes the data from `err` that corresponds to the trial you removed in RT (i.e., has the same index)
- prints the slowest RT remaining, rounded to two decimal places (*after* removing the outlier)
- prints the lengths of `rt` and `err` using a single `print` command, with accompanying text to make it clear which vale is the length of `rt` and which is the length of `err`

Note that this can be accomplished in 5 lines of code. However, you might start by figuring out how to do the task, in more lines of code, then figure out how to shorten your code if you can (think about nesting Python commands)

In [66]:
```
# Min should be 0.33, Size should be 23
deletion = rt.index(max(rt))
rt.pop(deletion)
err.pop(deletion)
print(round(min(rt), 2))
print('rt Length =', len(rt), ' &  err Length =', len(err))
```

Out[66]:
```
0.33
rt Length = 23 &  err Length = 23
```

Question 15

Print out all the values of RT, sorted from fastest to slowest. **Do not modify the original order of RT values** in doing this.

In [68]:
```
# w/verification
print(sorted(rt))
print('----------------------')
print(rt)
```

Out[68]:
```
[0.333356848, 0.389157455, 0.394252808, 0.401773569, 0.409225638, 0.435308188,
0.437877745, 0.442094359, 0.476159436, 0.479840688, 0.486154719, 0.495622859,
0.509801957, 0.518792127, 0.529661203, 0.534764366, 0.544626271, 0.565906723,
0.570404592, 0.573962432, 0.612854746, 0.714152493, 0.844916827]
----------------------
[0.394252808, 0.442094359, 0.534764366, 0.565906723, 0.570404592, 0.486154719,
0.518792127, 0.844916827, 0.495622859, 0.476159436, 0.612854746, 0.529661203,
0.389157455, 0.573962432, 0.714152493, 0.409225638, 0.435308188, 0.509801957,
0.544626271, 0.437877745, 0.333356848, 0.401773569, 0.479840688]
```

In [0]:
```

```