Lab 6

Table of contents

Beyond the single trial **3**
 Lab class . . . 3

Conditions files **6**
 Lab class . . . 6

PsychoPy loops **8**
 Lab class . . . 8
 Self-study . . . 8
Loop properties . . . 8
Adding and removing loops . . . 11
Confirmation . . . 11

Changing component properties **12**
 Lab class . . . 12
The basic idea . . . 12
Changing text . . . 12
 Self-study . . . 14
Changing images . . . 14
Changing locations . . . 15
Changing other properties . . . 15
Confirmation . . . 15

Formative PsychoPy quiz **16**
 Self-study . . . 16

Formative PsychoPy assignment **17**

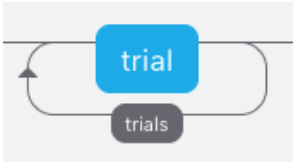
Explore, apply, reflect **18**
 Lab class/ Self-study . . . 18
Main exercise . . . 18
Challenge exercise . . . 19
References . . . 21

Beyond the single trial

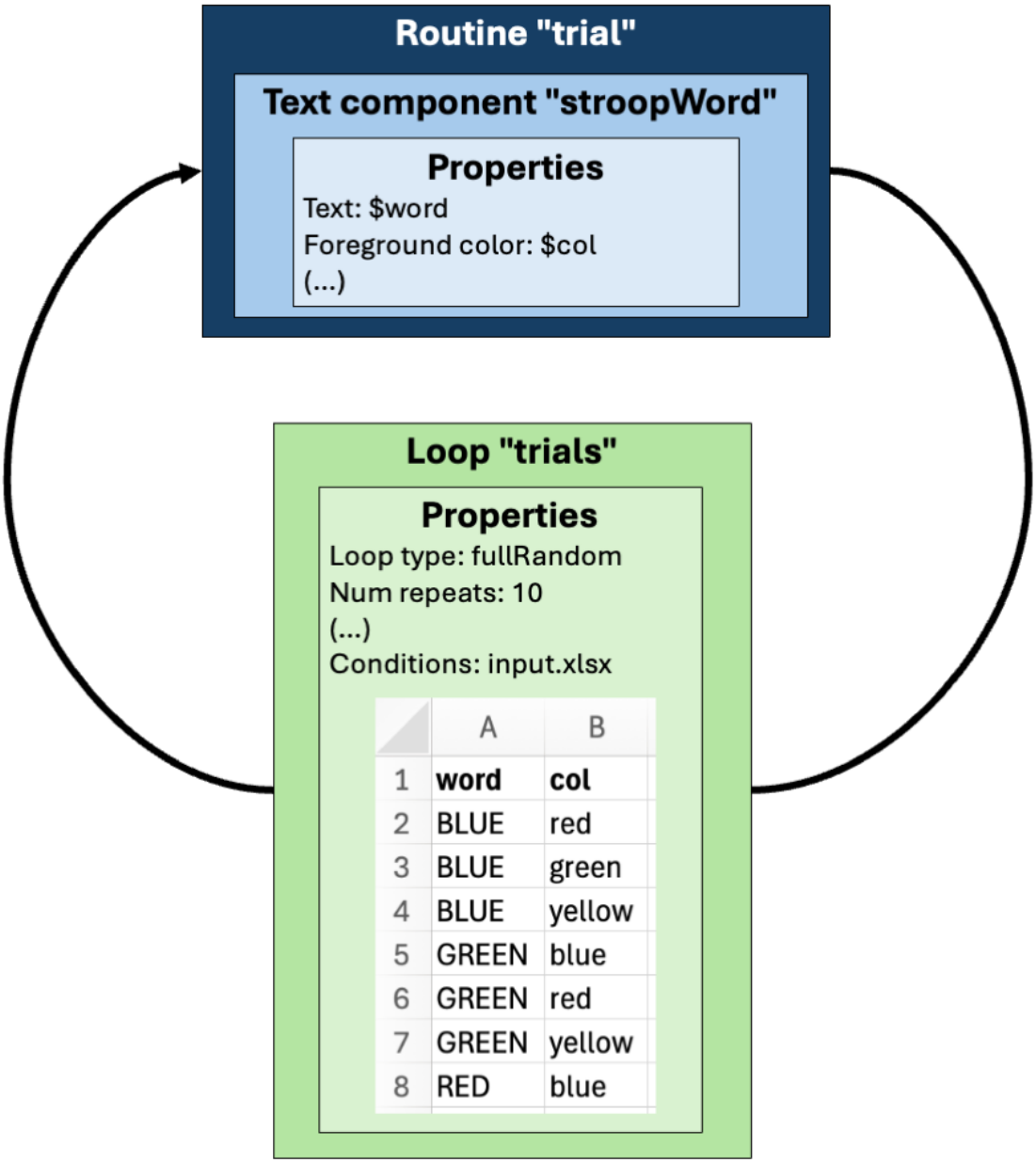
Lab class

This week, you will learn how to present multiple trials in PsychoPy using loops and conditions files. Our previous exercises focused on running a single trial. However, in a typical experiment, we have of course more than one trial. In theory, we could define a new routine for every individual trial:

However, given that experiments can include hundreds of trials, it would be extremely time-consuming and laborious to set up such an experiment. This is where loops come in: A loop repeats one or more routines a specified number of times. In this example, the loop `trials` repeats the routine `trial`:



Most importantly, when repeating routines in a loop, you can update component properties (e.g., stimulus size, colour, position) from one trial to the next. The information on how to update the component properties comes from a conditions file attached to the loop. Using a Stroop trial as an example, you can think about the relationship between conditions files, loops and modifiable component properties in the following way:



In this example, the routine `trial` has a Text component `stroopWord` that has properties (the text displayed and the text colour) that are variable (indicated by the $-sign—more about this later). The loop with its attached conditions file has the information on how these component properties should be changed from trial to trial (i.e., which word should be presented and what colour it should be presented in).

Warning

One of the most frequent errors made by PsychoPy beginners is this: If you want to change a component property (a piece of text, an image,...) in a routine from one trial to the next, this property can't be fixed. It must be defined using a variable.
If, for example, you want to present one image on the screen and change this image from one trial to the next, you can't set up the image component using a particular image (i.e., using a file name such as `cat.png`).
You also can't use two image components in the same routine, one presenting `cat.png` and the other presenting `dog.png`. Using two image components in the same routine would always present both images. However, on any given trial, you just want *one* image.
Therefore, the image must be a variable (e.g., `$img`).

Conditions files

Lab class

As mentioned in the previous chapter, we use **conditions files** (also referred to as **input files**) to tell PsychoPy what to change from one trial to the next. A conditions file will typically be an Excel file.[1]

Let's look at an example. Please open the conditions file for the letter flanker task from the previous lab (called `letter_flanker_input.xlsx` and located in the same folder as the `.psyexp` file). This spreadsheet has seven rows and three columns:

	A	B	C
1	stim	corrAns	congruency
2	HHHHH	c	con
3	SSSSS	n	con
4	SSHSS	c	incon
5	HHSHH	n	incon
6	BBHBB	c	neutral
7	BBSBB	n	neutral

The first row is a header row. Please note that the **naming conventions** for routines mentioned in **?@sec-naming-routines** also apply to the header row of your conditions file:

- Use only letters, numbers, and underscores.
- Always begin with a letter.
- Keep in mind that PsychoPy is case-sensitive: `stim` and `Stim` are two different things, as are `corrAns` and `corrans`.

The rows below the header row define all stimuli that you would like to present in your experiment (plus information associated with these stimuli such as correct responses). PsychoPy refers to these rows as **conditions**.[2]

The columns are what PsychoPy calls **parameters**. In our flanker conditions file, we have three different parameters: `stim`, `corrAns` and `congruency`. Parameters represent information about the conditions. For example, the parameter `stim` will determine which letters are presented on the screen, the parameter `corrAns` will determine what counts as a correct answer and the parameter `congruency` will determine if the trial is considered congruent, neutral or incongruent.

[1] You can also use .csv files.

[2] Please note that this does not mean that these necessarily correspond to the conditions in your experiment. In our flanker conditions file, we have six rows defining stimuli, but we would usually say that we have three experimental conditions (congruent, neutral and incongruent).

Note that the congruency information is not required for running the experiment, but it will be useful when analysing the data. Also, it allows you to easily check how many trials of each experimental condition you have in our conditions file. Note how this also determines the relative frequencies of experimental conditions: No matter how often we repeat the conditions file in a loop, 1/3 of trials will be incongruent, 1/3 congruent and 1/3 neutral.

PsychoPy loops

Lab class

Loops repeat routines within the loop. To learn more about a loop, you need to click on the name of the loop (e.g., `trials`). This will open the loop's property window. Details are explained below—for now the key information is:

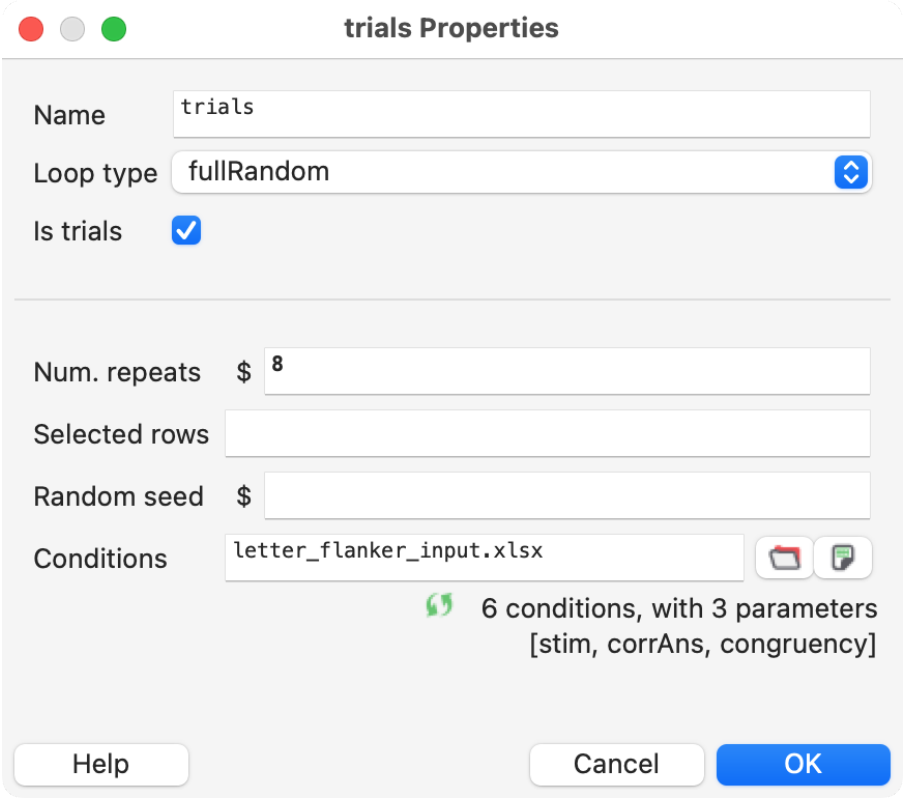
- **Conditions** has the file name of the conditions file.
- **Num. repeats** defines how often the rows in the conditions file should be repeated.
- **Loop type** determines if the rows in the conditions file are randomised.

Self-study

Conditions files must be attached to loops. Note that multiple loops can make use of the same conditions file. For example, the flanker experiment uses the conditions file `letter_flanker_input.xlsx` for both the loop `practiceTrials` and the loop `trials`.

Loop properties

Click on the loop `trials` in the flanker experiment to display the loop properties. Note that the column headers of the conditions file are shown in square brackets below the conditions/parameters information:



Here is what the fields in this properties window mean from top to bottom:

Name

The name of the loop. `trials` in our case.

Loop type

"Loop type" determines if PsychoPy keeps the order of non-header rows in the conditions file or randomises them. These are the options of interest to us:

sequential: The conditions file is not randomised. Trials are presented in exactly the same order as in the conditions file.

If we were to use 'sequential' for our conditions file, participants would always start with two congruent trials, followed by two incongruent trials, followed by two neutral trials. This is not what we typically want.[1]

random: The rows in the conditions file are randomised per loop repeat. Imagine you have four rows in your conditions file (the conditions A, B, C, and D) and you ask for two repeats ("Num. repeats"—see below). This is what PsychoPy will do:

[1] This option is there as sometimes we pre-randomise files, for example to exclude stimulus repetitions. If you pre-randomised a file, you do want PsychoPy to present trials in exactly the order they are in in the conditions file.

- First repeat
  - Take the four conditions and randomise them.
  - Example result: `C, D, A, B`
- Second repeat
  - Take the four conditions and randomise them.
  - Example result: `A, D, B, C`
- Trial order presented to the participant: `C, D, A, B, A, D, B, C` (note how the conditions are presented in groups of four).

fullRandom: The rows in the conditions file are multiplied by the number of repeats and are then randomised. Again, imagine you have four rows in your conditions file (the conditions A, B, C, and D) and you ask for two repeats. This is what PsychoPy will do:

- Multiply conditions by number of repeats: `A, B, C, D` * 2 = `A, B, C, D, A, B, C, D`
- Randomise *all* conditions.
- Example result: `C, D, A, A, B, C, D, B`
- Note how the four conditions are no longer grouped together.

There is only a difference between 'random' and 'fullRandom' if the number of repeats is more than 1. If 'random' or 'fullRandom' is the better choice depends on what you would like to achieve. With 'random', you know that your conditions will be evenly distributed across the experiment (because they are presented in groups). You can also reduce the number of stimulus repetitions, if that is a potential issue in your experiment (this is because conditions can only repeat if they happen to be at the end of one group of four and at the beginning of the next group of four). On the other hand, your participants might realise that stimuli are presented in groups of four and they might start to predict what is going to happen next. This would be a confound. Therefore, I tend to go for 'fullRandom' if the number of conditions is small (say, around ten conditions) and for 'random' if it is larger.[2]

For both 'random' and 'fullRandom', the randomisation will always randomise *complete* rows. That is, if a row includes the information `HHSHH`, `n` and `incon`, these parameters will always stay together.

Is trials

Not relevant for us. Simply leave this ticked.

Num. repeats

How often the conditions file should be repeated. In our flanker task, there are 8 repetitions of 6 conditions, so 48 trials overall.

Note: Setting "Num. repeats" to 0 will skip all routines in the loop. This can be useful during task development.

[2] If you have more than two repeats, say four, an alternative would be to combine the two approaches. That is, you could have `A, B, C, D, A, B, C, D` in your conditions file and then opt for `random` and two repeats.

Selected rows and Random seed

Not relevant for us.

Conditions

The name of the conditions file (including the path[3]). To add a conditions file to a loop, click on the folder icon, navigate to the file and add it to the loop.

Adding and removing loops

To add a new loop to an experiment, you need to click on "Insert Loop" in the flow panel (or on Experiment "Insert Loop in Flow" in the menu bar). Then, click on the timeline where you want the loop to begin.

- If there is just one routine in the experiment, this will automatically bring up the property window discussed above. You can then edit the relevant properties and click on OK.
- If there are multiple routines in the experiment, PsychoPy expects you to click on the location where you want the loop to end first. Once you have done so, the properties window will appear.[4]

To remove a loop, right-click on the loop and then click on "remove".

Confirmation

> **!** Important
>
> Please confirm you have worked through this chapter by submitting the corresponding chapter completion form on Moodle.

[3] PsychoPy will automatically add the correct path for you if you click on the folder icon to add the conditions file, provided you have saved the experiment before adding the conditions file. If conditions file and experiment file are in the same folder, only the file name is required.

[4] Sometimes it happens that the loop endpoint on the timeline moves when you move the mouse pointer to the properties window. If this happens, you have to delete and reinsert the loop. If you move the mouse pointer straight up from where you clicked, you can avoid this happening.

Changing component properties

Lab class

The basic idea

How do you tell PsychoPy that you want to change a component property (e.g., a piece of text, an image or a stimulus location) from one trial to the next? You tell PsychoPy that the property is variable.

How do you tell PsychoPy that the component property is variable? You provide PsychoPy with a variable name and set the component property to "set every repeat".

Frequently, the variable name will come from the header row in your conditions file.[1] It is probably most instructive to have a look at a more detailed example of this idea. We will use the Text property of the Text component for explaining the basic idea in more detail. Other properties will then be dealt with more briefly.

Changing text

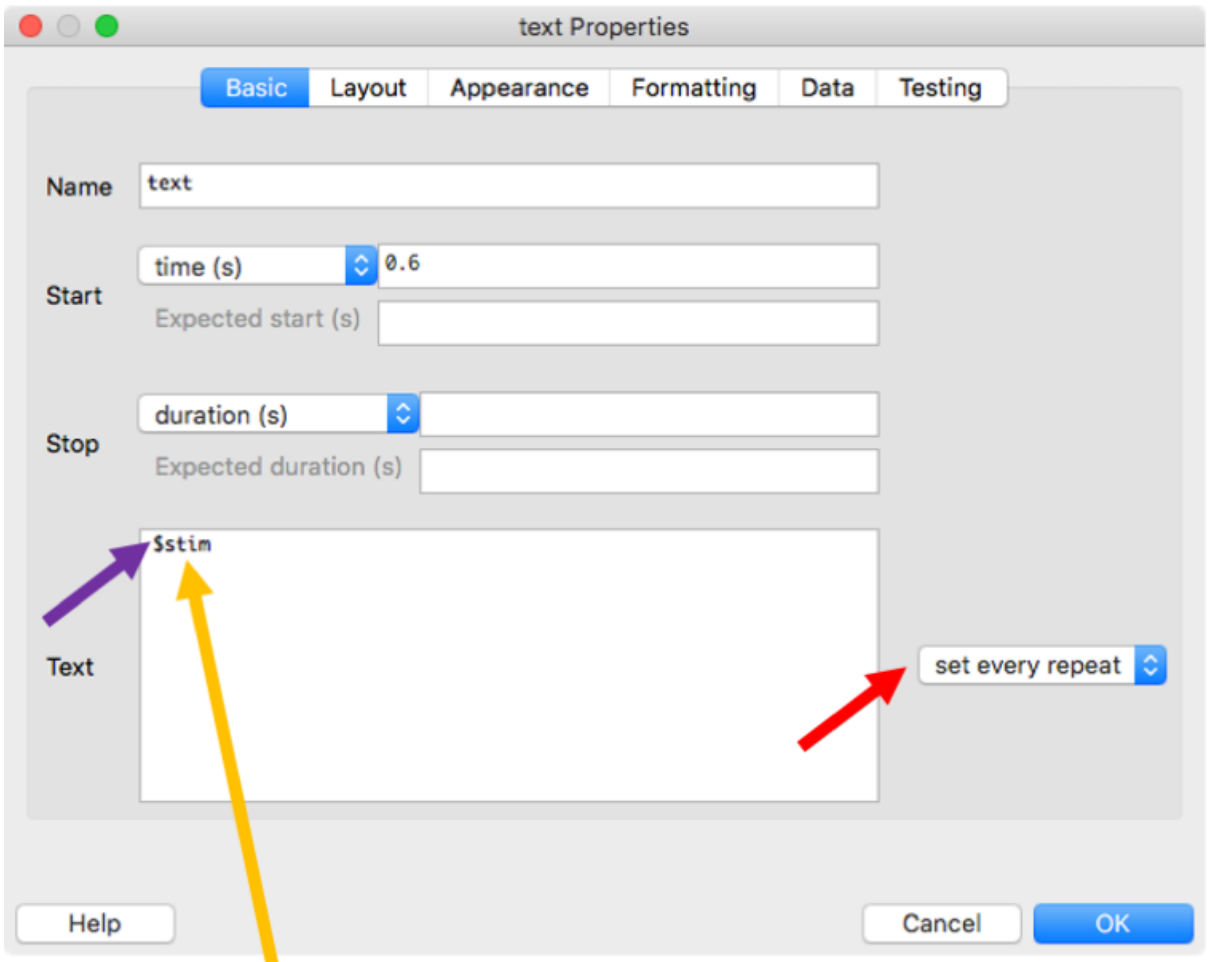
If your aim is to change a piece of text from one trial to the next, you must update the Text property of the Text component. For this to work, you need:

- A variable name in the Text property field that is preceded by a $ sign.[2]
- An conditions file that has a column header that matches this variable name exactly.
- To select "set every repeat".

These requirements are illustrated here:

[1] When you learn how to give feedback, you will see that you can also use Python code to create variable names.

[2] Why is there a $ sign before the variable name? This tells PsychoPy that you don't mean to literally present `stim` on the screen, but that `stim` should be interpreted as a variable name.



	A	B	C
1	stim	corrAns	congruency
2	HHHHH	c	con
3	SSSSS	n	con
4	SSHSS	c	incon
5	HHSHH	n	incon
6	BBHBB	c	neutral
7	BBSBB	n	neutral

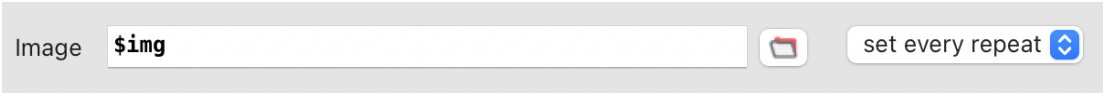
> **Tip**
>
> As this is such a common mistake, let us reiterate that the variable name in the Text field and the header in the conditions file need to match **exactly**. For example, the experiment will not run if you have `stim` in the Text field and `Stim` in the conditions file!
> A frequent error that can be hard to spot is a space after the column header in Excel, e.g., `stim_` (where _ represents a space). The most straightforward way to identify this issue is to open the conditions file in Excel and to right-align the contents of the columns.

Note that sometimes you might need more than one column in the conditions file to update stimulus properties. For example, to update the Stroop stimulus RED, you need to get information about the word as well as the colour from the conditions file. Thus, your conditions file needs separate columns for defining word and colour (see my Stroop experiment in **?@sec-stroop-task**). The same is true when you would like to change, say, the stimulus and its location from trial to trial.

Self-study

Changing images

If your experiment uses images that change from trial to trial, you will need to use an Image component and tell PsychoPy that the image should be updated on every repeat. The basic idea is the same as before: You provide PsychoPy with a variable name preceded by a `$` sign and select "set every repeat":



Again, the variable name needs to match a column header in the conditions file:

	A
1	img
2	img1.png
3	img2.png
4	img3.png
5	img4.png
6	img5.png
7	img6.png

Note that the file name needs to include the extension of the file (`.png` in this example). Also, when defined without a path, the images must be located in the same folder as the `.psyexp` file.

Changing locations

If your experiment requires the location of stimuli to change from trial to trial, you will need to use the Position property of your component. The basic idea is the same as before: You provide PsychoPy with a variable name and select "set every repeat". For example, if you would like to change the stimulus position along the y-axis, you could achieve this in the following way:

Again, the variable name needs to match a column header in the conditions file:

	A
1	yPos
2	0.25
3	0
4	-0.25

You might have noticed a difference to before though: We did not include the $ sign. Why is this? Note that next to "Position [x,y]" there already is a $ sign. This indicates that you should not add a $ sign here.[3]

What is the logic behind this? You do not need to add the $ sign when it does not make sense to enter text into a given field. For example, the position along one of the axes can only be a number. If you enter text, it is clear that this is meant to be a variable name. On the other hand, other properties *can* be represented by text (e.g., colours or file names). In these cases, you need to add the $ sign to indicate that you would like PsychoPy to interpret the input as a variable name.

Changing other properties

All other properties that can be changed from trial to trial (e.g., size, colour or orientation) work analogously to the examples illustrated above.

Confirmation

> **!** Important
>
> Please confirm you have worked through this chapter by submitting the corresponding chapter completion form on Moodle.

[3] In fact, doing so would stop the experiment from running.

Formative PsychoPy quiz

Self-study

This is a formative PsychoPy quiz about content covered in Lab 6. You might find it helpful to return to these questions when revising for the January exam.

This section contains interactive content which is not available in the PDF version. Please visit the online version to see it.

Formative PsychoPy assignment

The instructions for the formative PsychoPy assignment will be available in the Quizzes and Assignments section on Moodle from **Thursday, 13 November from 3pm onwards**. Please note that the instructions also explain how exactly the assignment needs to be submitted.

Please submit your experiment by **Thursday, 20 November at 3pm**. Please note that we will provide you with feedback on issues commonly observed in the submitted experiments. Due to time constraints, it will not be possible to provide you with individual feedback (you will receive individual feedback on your summative PsychoPy assignment submission though).

This is a formative assignment that will not contribute to your overall mark. Formative assignments not submitted by the deadline incur no penalty. While formative, we would highly recommend that you complete the assignment. Successfully completing the formative assignment will help you to do well on the summative assignment: In previous years, we found that students who submitted a formative PsychoPy assignment performed significantly better on the summative assignment compared to those students who did not submit a formative PsychoPy assignment.[1]

Please submit your formative PsychoPy assignment even if it is not quite working or doesn't run at all. This will give us a better idea of the issues you encountered when creating the experiment.

Assignment feedback

Download the feedback presentation.

[1] Note that this is a great example of self-selection though! Students were not randomly allocated to conditions, but decided themselves whether or not to complete the formative PsychoPy assignment. That said, we probably don't need to run an experiment with random allocation to convince you that practising using PsychoPy will help when it comes to creating the summative PsychoPy experiment.

Explore, apply, reflect

Lab class/ Self-study

Please complete as many tasks as you can during the lab class. If you don't finish them all, complete the remaining ones as part of your self-study.

Main exercise

In this exercise, you will modify a choice reaction time task.

Click here to download the task.

Once downloaded, unzip the file and open the experiment using PsychoPy and the conditions file using Excel.

Task 1: Use an Image component

Remove the Text component in the routine `trial` and use an Image component instead. Use two different images in your experiment (replacing the previous stimuli "&" and "%" in the experiment you downloaded; you can use the two images from Lab 5 if you would like to). On each trial, one of the images should be presented.

Also, change the response keys to use the arrow keys on the keyboard.

Make sure to adapt the conditions file and the task instructions accordingly.

> How do I find out what the arrow keys are called in PsychoPy? (Hint: The response key names are not < and >!)
>
> PsychoPy comes with a number of helpful demos. Before using the demos for the first time, you need to unpack them (Demos Unpack Demos...). If you are using one of the iMacs, unpack them on the Desktop. If you are using your own machine, choose a location of your own liking.
> Once unpacked, open and run the demo `keyNameFinder`. When `keyNameFinder` runs and you press a key on the keyboard, the name of this key in PsychoPy will be displayed on the screen.

Task 2: Use a Polygon component

After having saved your previous exercise, make a copy for this exercise. Then, remove the Image component in the routine `trial` and use a Polygon component (located in Stimuli components) instead. Change the colour of the Polygon component from trial to trial.

Make sure to adapt the conditions file and the task instructions accordingly.

Challenge exercise

The 2-letter version of the letter flanker task has a potential confound: congruent stimuli always consist of exactly the same letters, whereas incongruent stimuli consist of different letters. Maybe the interference effect is due to the fact that the row of identical letters in congruent trials speeds up letter recognition, and not the fact that in incongruent trials flanker and target map onto different responses?

Investigate this question by creating a 4-letter version of the flanker task as a 4-letter version allows for congruent trials where flankers and target differ.

The four letters in your task and the correct responses are:

- B and G: left arrow key
- D and Q: right arrow key

For the purpose of defining correct responses in an conditions file, the names for the arrow keys are `left` and `right`.

Please create an experiment that fulfils the following criteria:

- Congruent and incongruent trials should be equally frequent.
- Congruent trials should *not* include strings of identical letters (e.g., BBBBB).
- There should be 64 trials overall.
- The conditions file should be repeated 4 times.
- The letter presentation should begin 0.5 seconds after the start of the trial.
- The presentation of the letters should end when a response key is pressed.

- Trials should be presented in random order.
- PsychoPy should write accuracy information to the output file.
- Trial-wise congruency should also be coded in the output file.

Your stimuli should be:

- 150 pixels high
- White on black background
- Presented in the centre of the screen

How to begin?

Start by opening my version of the flanker task the you downloaded in Lab 5 and try to work out how this task works. Run it. Play around with it. Change things and see what happens. As mentioned in the previous lab, you can't really break anything. If you change something and the task no longer works, simply download it again (or make a copy or the original version before you make changes to it).
Once you think you have a sufficient understanding of how the task works, adapt this two-letter version instead of creating a completely new experiment.

How do I set up the conditions file? (Hint 1)

The basic idea is to create all possible letter combinations that meet the criteria defined above.

I've read Hint 1, but I'm still stuck on the conditions file. (Hint 2)

You might be puzzled by the fact that you end up with fewer possible congruent stimuli (4) than incongruent stimuli (8). The solution is to simply duplicate the congruent stimuli, so you have eight congruent and incongruent stimuli.

Aaaarghhh!!! I've read Hint 1 and Hint 2, but I'm still stuck. (Hint 3)

These are your eight congruent stimuli:

```
BBGBB, BBGBB, GGBGG, GGBGG, DDQDD, DDQDD, QQDQQ, QQDQQ
```

These are your eight incongruent stimuli:

```
DDBDD, QQBQQ, DDGDD, QQGQQ, BBDBB, GGDGG, BBQBB, GGQGG
```

You now have 16 "conditions" in your conditions file. Thus, you need four repetitions to achieve 64 trials overall.

You can watch a video of the solution here:

This section contains content which is not available in the PDF version. Please visit the online version to see it.

References