Lab 7

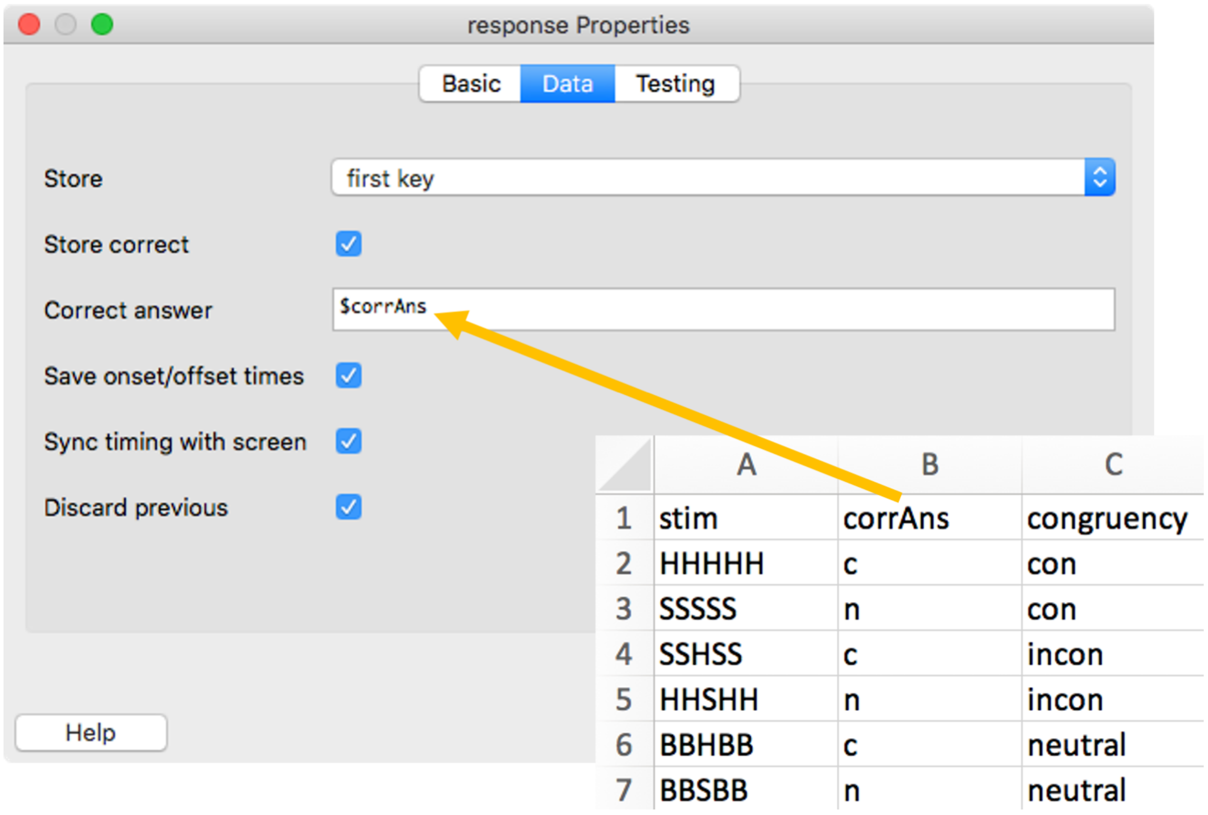
Table of contents

Accuracy and feedback

Lab class

Determining accuracy

A parameter in a conditions file can also be used to determine if the participant gave a correct response. Here is the Data tab of the Keyboard component from our flanker task combined with a screenshot of the conditions file:



Store	first key
Store correct	✓
Correct answer	$corrAns
Save onset/offset times	✓
Sync timing with screen	✓
Discard previous	✓

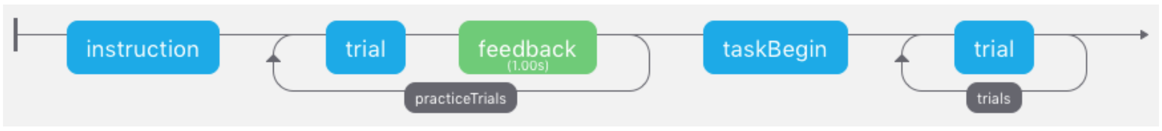
	A	B	C
1	stim	corrAns	congruency
2	HHHHH	c	con
3	SSSSS	n	con
4	SSHSS	c	incon
5	HHSHH	n	incon
6	BBHBB	c	neutral
7	BBSBB	n	neutral

The orange arrow illustrates how the column called `corrAns` in our conditions file is linked with our Keyboard component. Note that you need to tick “Store correct” for the field “Correct answer” to appear. Also note that the “Allowed keys” (on the Basic tab) can be constant as the allowed keys are the same on every trial.

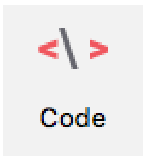
If “Store correct” is ticked and a variable name for “Correct answer” is provided, PsychoPy will automatically write information about the accuracy of responses to the output file it produces. However, this information is not just useful for the output file. It can also be used to give participants feedback about their performance.

Giving accuracy feedback

Sometimes you might want to give participants feedback on their performance. In particular, this might be the case while they are still practising the task, as shown in the flow of our flanker task:



In this example, participants will be presented with a routine called `feedback` after every practice trial. The problem is that you cannot know what type of feedback (e.g., correct or incorrect) will be required in advance, because the feedback will depend on the participant's response. Therefore, you will need to figure out what feedback to give on the fly. To achieve this, you need to add a **Code component** (located in **Custom** components) to the feedback routine:



Code components allow you to insert Python code into an experiment. You don't need to learn how to write Python code to be able to do this, but you need to know a few Python basics. I've compiled these basics in **?@sec-python-basics**.

The basic idea that underlies giving accuracy feedback is this:

- You have a routine where responses can be either correct or incorrect (usually, your trial routine).
  - In the Keyboard component of this routine, you tell PsychoPy to store whether or not a response was correct.
  - PsychoPy automatically codes **correct responses as 1**, and **incorrect responses as 0**.
- In a subsequent routine, you use the information that PsychoPy has stored on the previous routine to provide the participant with feedback.
  - If the response was correct, you present positive feedback.
  - If the response was incorrect, you present negative feedback.

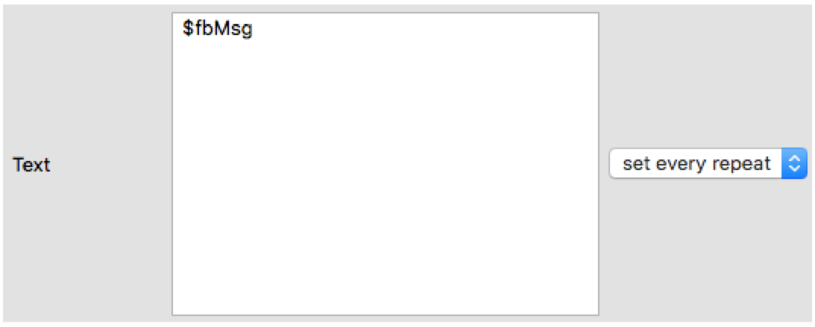
Let us go through how this works in our letter flanker task. Please open the flanker task from Lab 5 in PsychoPy and have a look at the relevant routines and components:

- Routine `trial`
  - The Text component presents the stimulus.
  - The Keyboard component records the response. If we told it to, the Keyboard component will also store the accuracy of the response. ▯ This is the information we need to give accuracy feedback!
- Routine `feedback`
  - The Code component checks whether the response was correct or incorrect.

  - Depending on the accuracy, the Code component updates the message to be shown as feedback.
  - The Text component presents the feedback message.

Here is what we need to know to make this work:

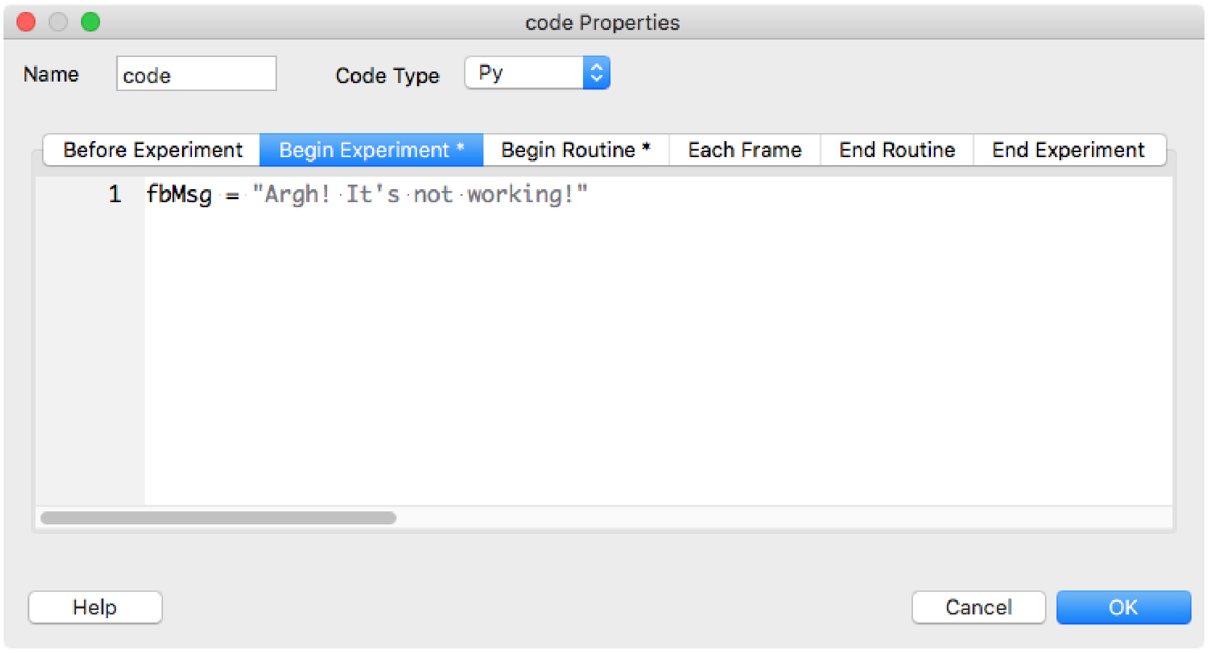
- The name of the Keyboard component that stores the accuracy of the response. In our flanker task, the name of the Keyboard component is `response`.
- PsychoPy stores the accuracy information in a variable called `<nameOfResponseComponent>.corr`
  - The text between the angle brackets is a place-holder for the actual name of the component.
  - In our example, the relevant variable would be `response.corr`.
- The feedback message itself is some text that needs to change from trial to trial. Therefore, it needs to be variable. In our flanker task, the variable is called `fbMsg`.
- After the Code component has figured out what the feedback message should be, we would like to present the feedback to the participant using a Text component. We thus need to add `$fbMsg` to the Text component presenting the feedback in the routine `feedback` and set it to `set every repeat`.


Text $fbMsg set every repeat

Now, let's look at the actual Python code. We need two pieces of code in two different places. The first piece of code is this (you can simply copy and paste the code into your experiment!):

```
fbMsg = "Argh! It's not working!"
```

This needs to be placed in the "Begin Experiment" tab of the Code component:



```
fbMsg = "Argh! It's not working!"
```

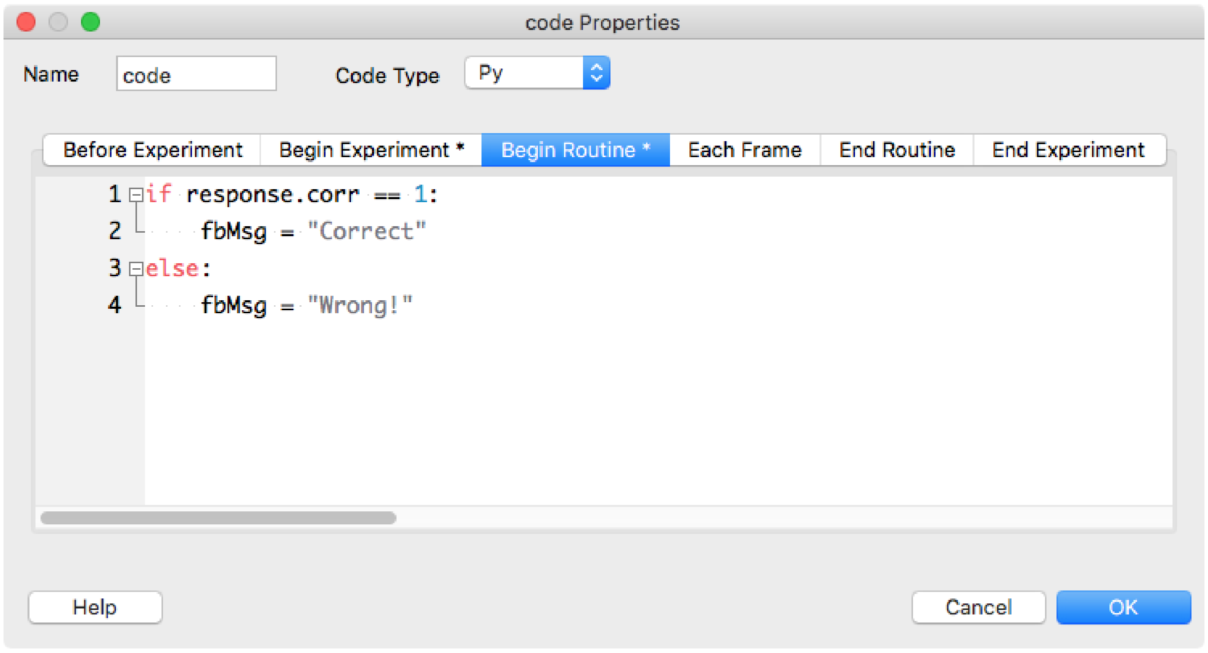
The aim of this is to define the variable and to assign a value to it that will help you figure out whether or not your feedback is working. You could simply define an empty string, but this will make it harder to determine what is going wrong if the feedback doesn't work as intended:

```
fbMsg = ""  # possible, but not useful!
```

The second piece of code is this:

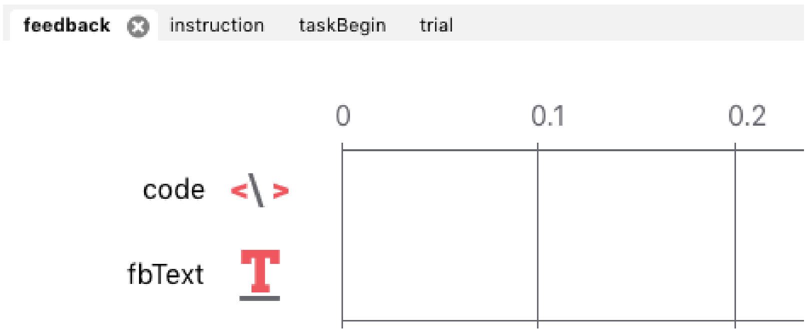
```
if response.corr == 1:    # if the response was correct
    fbMsg = "Correct!"    # this should be our FB message
else:                     # alternatively (i.e., the response was incorrect)
    fbMsg = "Incorrect!"  # this should be our FB message
```

This needs to be placed in the “Begin Routine” tab of the Code component:



```python
if response.corr == 1:
    fbMsg = "Correct"
else:
    fbMsg = "Wrong!"
```

Note that the order of components is important! In the routine feedback, the Code component needs to be *above* the Text component, otherwise the feedback will not work (see Section):



Self-study

Too slow feedback

For some experiments, you might also want to give "Too slow!" feedback if a participant does not respond before a set RT deadline.[1] In this case, you need to slightly modify the above approach, because PsychoPy codes both "wrong" responses and "too slow" responses as `0`. Thus, we also need to check whether or not a key was pressed to decide if the feedback should be "Too slow!" or "Wrong!". This is the code:

[1]Note this requires that your trial has a defined Stop time. Otherwise it will never end and participants can not be too slow.

```
if response.corr == 1:  # correct response
    fbMsg = "Correct!"
elif response.corr == 0 and response.keys != None:  # incorrect response given
    fbMsg = "Wrong!"
elif response.corr == 0 and response.keys == None:  # no response given
    fbMsg = "Too slow!"
```

How does this work?

- PsychoPy stores the information about pressed keys in a variable called `<nameOfResponseComponent>.`
- `==` means `is equal to` and `!=` means `is not equal to`.
- In Python, `None` (capitalisation is again important here!) means "no value"; here it means that no response keys were pressed.

You could replace the second `elif` statement by simply saying `else:`, but being more verbose sometimes makes it easier to understand what the code does.

Confirmation

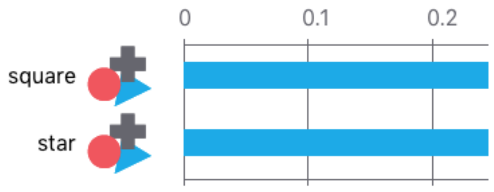
! Important

Please confirm you have worked through this chapter by submitting the corresponding chapter completion form on Moodle. :::# Miscellanea {#sec-miscellanea}

Self-study

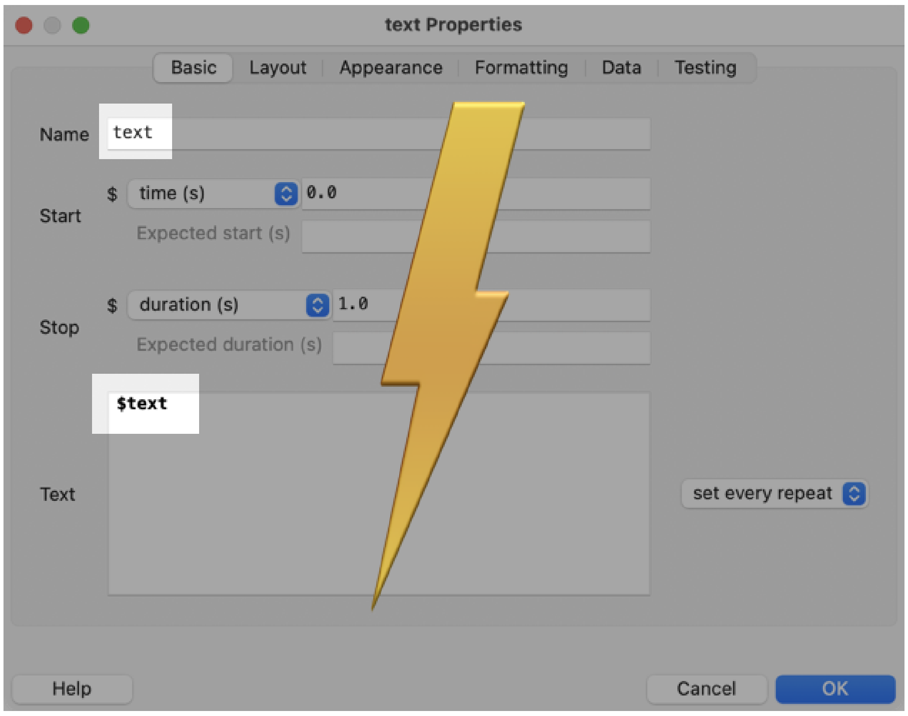
PsychoPy processes components from top to bottom

Within a routine, PsychoPy processes components from top to bottom. If you would like to draw a star[2] on top of a square, the square needs to come first, followed by the star:



Names must be unique

In addition to following the naming restrictions described previously, you must also make sure that all names are **unique**. For example, if you have a Text component whose name is `text`, you cannot also use a variable called `$text`.


text Properties

Basic | Layout | Appearance | Formatting | Data | Testing

Name: text

Start: $ time (s) 0.0
Expected start (s)

Stop: $ duration (s) 1.0
Expected duration (s)

Text: $text — set every repeat

This restriction applies across all routines, components and variables of an experiment. For example, if you're using `$stim` in any one routine, you will not be able to use that variable name in any other routine, no matter where in your experiment it appears.

Skipping routines

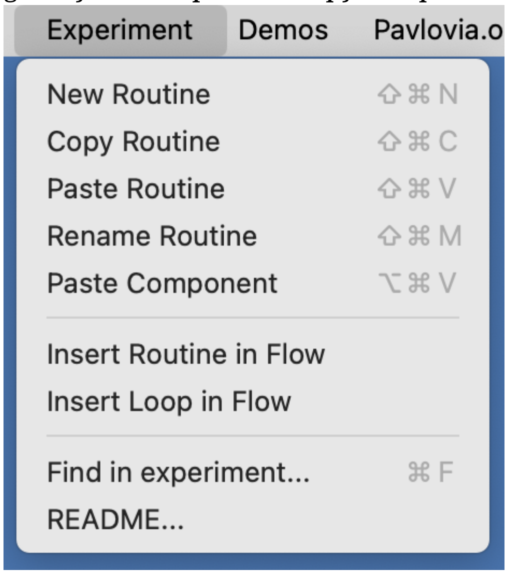
Sometimes you might want to *not* present a routine. In PsychoPy, this can be done by using `continueRoutine` in a code component. For example, you might want to present "Too slow!" feedback only if participants took more than 1.5 s to respond. This can be achieved with a feedback routine that has the following code component in the "Begin routine" tab:

```
if response.rt > 1.5:  # if the response was slower than 1.5 s
    continueRoutine = True  # present the feedback routine
else:
    continueRoutine = False  # do not present the feedback routine
```

Below the code component, the feedback routine would also need a Text component that displays the text "Too slow!" if the routine is presented. This text could be set to `constant` as it never changes.
For testing purposes, it is often convenient to **skip multiple routines** (e.g., if you don't want to run the practice trials, and instead want to run the main trials straight away). This can be achieved by adding a loop around these routines and setting "Num. repeats" for this loop to `0`.

Copying and pasting routines and components

In **?@sec-routines**, we briefly mentioned that the “Experiment” menu in the menu bar gives you the option to copy and paste routines:



To copy a routine, click on the routine’s tab (if it is not already selected), then click on “Experiment” and “Copy Routine”. Finally, click on “Experiment” and “Paste Routine”. It is also possible to copy components. To copy a component, right-click on the component and then click on “copy”. Then click on the routine into which the component should be inserted, and click on “Experiment” and “Paste Component”.
The way to copy and paste routines and components across experiments is to open two Builder windows. Here is how to do this:

- Open the first experiment as usual.
- Then click on File ▯ New to open a second Builder window.
- Then click on File ▯ Open to open your other experiment in the second Builder window.

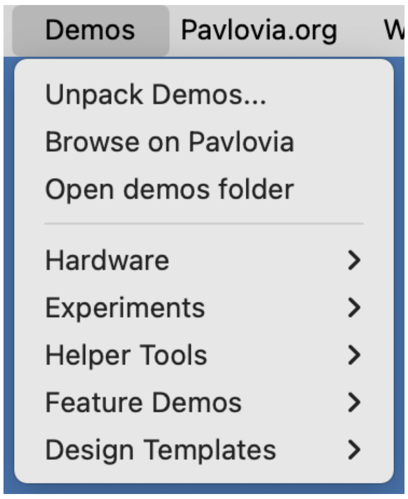
Please make sure to follow these exact steps!

Windows users

In the past, copying/pasting routines under Windows would not work if you opened the second experiment by double-clicking on the .psyexp file in the File Explorer. It was necessary to use File ▯ New instead. As I do not have a Windows operating system, I am not sure if that is still the case.

PsychoPy demos

PsychoPy comes with a number of demos included. To see the demos, you need to click on “Demos” in the menu bar and then on “Unpack demos”. Next, you need to choose a location on your computer where the demos should be unpacked (this will create a new folder “PsychoPy3 Demos”). If you now click on “Demos” in the menu bar again, a list of demos will be displayed:



A useful demo is the "keyNameFinder" (in "Helper Tools"). It displays key names on the screen. If you would like to use a particular key in your input file, but you are not sure what its name is, run this demo.
The demo "spatialUnits" (also in "Helper Tools") is helpful for understanding PsychoPy's spatial units (height, norm, pix, etc.). This demo can be used in addition to our interactive app for visualising PsychoPy spatial units.

PsychoPy problems and errors

?@sec-psychopy-issues includes a checklist for evaluating your experiment and covers frequently encountered PsychoPy error messages.

Confirmation

Important

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

Formative PsychoPy quiz

Self-study

This is a formative PsychoPy quiz about content covered in Lab 7. 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.

Food for thought: Validity

Self-study

I recently came across an article with the headline Britain one of least 'nature-connected' nations in world – with Nepal the most. After reading it, I thought it was a good opportunity to think in a bit more depth about validity, specifically **content validity**.
Here are some quotes from the article:

> Nature connectedness is a psychological concept that measures the closeness of an individual's relationship with other species. **Researchers found the strongest indicator for a close relationship with nature was high levels of "spirituality" in a society.** More religious societies and cultures where there was a preference for faith over science showed high levels of nature connection.
>
> The correlation between nature connection and "spirituality" in countries was discovered using measurements of the importance of religion, beliefs in a god and faith in different countries recorded by the World Values Survey.
>
> "Nature connectedness is not just about what we do, but how we feel, think, and value our place in the living world," said Richardson, who admitted he was not surprised that Britain languished so low in the nature connection league table. "We've become a more rational, economic and scientific society," [Richardson] said. "How do we reintegrate natural thinking in our very technological world? How do you create sacred urban nature?"

As someone who cares about the environment, but who is not a particularly spiritual or religious person, I was curious about the correlation between nature connectedness and spirituality. So, how was "nature connectedness" operationalised in this study (Richardson et al., 2025)? It was defined as the latent mean[1] on a shortened version of the connectedness to nature scale (Mayer & Frantz, 2004). The following items were rated from 1 (strongly disagree) to 5 (strongly agree):

1. I feel as though I belong to the Earth as equally as it belongs to me.
2. I think of the natural world as a community to which I belong.
3. I often feel part of the web of life.
4. Like a tree can be part of a forest, I feel embedded within the broader natural world.
5. I feel that all inhabitants of Earth, human, and nonhuman, share a common 'life force'.
6. When I think of my life, I imagine myself to be part of a larger cyclical process of living.

7. I often feel a kinship with animals and plants.

Looking at these items, it seems to me that to strongly agree with most of these items, one needs to hold beliefs that are aligned with animistic, vitalistic, romantic, Buddhist and/or pantheistic ideas. Considering this, a correlation with spirituality is perhaps not particularly surprising.

What does that mean in terms of validity? One could argue that this is simply what nature connectedness *is*—a fundamentally spiritual belief. From this perspective, the scale has strong content validity because the items accurately reflect the construct as defined.

While this view is internally consistent, I believe it's potentially problematic. The article frames low nature connectedness as a negative trait, implying that societies should move towards a more "spiritual" worldview.

I'm not sure that's desirable or necessary. Surely, it must be possible to recognise our connectedness to nature in a more rational and scientific way. It must be possible to hold a non-spiritual worldview and still recognise our profound dependence on the environment.

I asked Gemini to generate some items that might represent a more scientifically based nature connectedness. Here is what it came up with:

1. I recognise that the human species is fundamentally dependent on the health and stability of global ecosystems.

2. I conceptualise the human species as one component within a complex, interdependent ecosystem.

3. I think that the ongoing degradation of the natural environment poses a significant threat to the long-term security and well-being of the human species.

4. I recognise that all known organisms are connected through shared, fundamental biochemical and evolutionary principles.

If we accept the premise that there could be spiritual and non-spiritual bases for nature connectedness, the connectedness to nature scale in its current form appears to suffer from **construct underrepresentation**. Content validity requires that the items be a representative sample of the *entire* content domain of the construct. If there are other, equally valid ways to be "connected to nature" that are not represented by these items, then the scale's content validity is compromised. This highlights the importance of critically examining not just the results of a study, but also the methods and tools used to generate them.

Explore, apply, reflect

Lab class/ Self-study

Your task is to extend the choice reaction time task from Lab 6. You should include 16 practice trials. On these practice trials, the participant should receive feedback on the accuracy of their response. Make sure your feedback works by running the task and making correct as well as incorrect responses.
If you have no previous experience using Python, we would recommend reading **?@sec-python-basics** before starting to work on the exercise.

How do I go about this?

Adapt the feedback from the flanker example we shared with you in Lab 5. The main thing you need to keep in mind is that you need to adjust variable names. For example, the Keyboard component in the flanker task is called `response`. If your Keyboard component is called, say, `key_resp` you must adapt the code accordingly.

References

Mayer, F. S., & Frantz, C. M. (2004). The connectedness to nature scale: A measure of individuals' feeling in community with nature. *Journal of Environmental Psychology*, *24*(4), 503–515. https://doi.org/10.1016/j.jenvp.2004.10.001

Richardson, M., Lengieza, M., White, M. P., Tran, U. S., Voracek, M., Stieger, S., & Swami, V. (2025). Macro-level determinants of nature connectedness: An exploratory analysis of 61 countries. *Ambio*. https://doi.org/10.1007/s13280-025-02275-w

[0] You can draw simple geometrical shapes in PsychoPy using the Polygon component in the "Stimuli" components section.

[0] Instead of just averaging raw survey answers (which can be skewed by things like translation issues or random error), a latent mean estimates the group's "true" underlying score by correcting for imperfections in the questions. This makes comparisons between different cultural groups more accurate.