Documentation/Help

Choose from a menu of all object types and documentation

pb

Open documentation for an object type (all properties)

pb <object type> e.g. `pb cross` (or can be shorter: `pb cr`)

or input objects directly:
pb(objects)

See name and default value for a property

pb <object type>.<property> e.g. `pb cross.size` (or can be shorter: `pb cr.si`)

Preferences

Set preferences: default property values for trial, experiment, screen, other device objects

pb_prefs

Switch to compatibility instead of precise visual timing

pb_prefs → screen → disable sync tests and/or enable system compositor

Making and running an experiment

Set up an experiment spreadsheet file **(optional)**

makeExperiment

Experiment spreadsheet syntax

^	– Fill cells in a column from a vector/array
*#	– Repeat cells in a column # times
*	– Generate combinations across columns
*A[b]	– Generate combinations across columns in group *A*, keeping cells in group *b* together
?	– Randomize order in a column
?#[b]	– Randomize order in groups of # in columns, keeping cells in group *b* together across columns
$	– Repeat value, not expression
-	– Leave property/target unset

Randomization

See also *rep*, *randomNum*, *randomNum_normal*, *randomRoll*, *randomChoose*, *randomOrder*, *randomBalancePerms*, etc. in *<PsychBench folder>/tools*.

Load an experiment from a spreadsheet file into memory in MATLAB

loadExperiment

View the experiment in memory in table form

viewExperiment
viewExperiment –d – sort by trial definition

Run the experiment in memory

runExperiment

Quit and resume an experiment

Ctrl + *Esc* to quit
saveExperiment → *.mat* file
loadExperiment ← *.mat* file
runExperiment

Show elements without needing to make an experiment

objects(s) = **<type>Object**([number of objects])
showElements(object(s), … , [options …])

Coding method (optional)

newExperiment

For each trial…
element/trial objects(s) = **<type>Object**([number of objects])
addTrial(objects(s), … , [trial definition group # / name], [trial number])

setTrialList(trial list)

objects(s) not specific to trial = **<type>Object**([number of objects]) …
addToExperiment(object(s), …)

Trial timing

Start a trial whenever previous trial ends + pre-trial interval.
End a trial when no elements are left running or scheduled to start.

(This is the default.)

Start a trial at fixed time from a trigger used to sync the experiment in a past trial

`trial.start.t_sync = [a b]`
a = start time relative to sync if this is the first trial that runs after it (sec)
b = start time increment if this is a later trial (sec)

See below for triggers/sync.

Pre-trial interval

`trial.preTrialInterval =` interval (sec)
Default: 0.75 sec

Element timing

Start/End an element at time from trial start

```
element.start.t = time (sec)
element.end.t   = time
```

`element.start.t = 0` → start at trial start

End an element at duration

```
element.end.duration = time (sec)
```

Start/End an element at other element start/end

```
element.start/end.startOf = string(s) pointing to other element(s) in the trial
element.start/end.endOf   = string(s)
```

e.g.

```
 "picture"
 "pictures(2)"
 "pictures(" + 2 + ")"
["pictures(2)" "pictures(3)"]
```

Start/End an element at any response recorded by a response handler element

```
element.start/end.response = true
```

See also *start/end* field *responseBy*.

Start/End an element at time from a trigger used to sync the experiment in the same trial

```
element.start/end.t_sync = time (sec)
```

See below for triggers/sync. See also *start/end* fields *trigger*, *triggerBy*.

Run an element in the pre-trial interval

```
element.start.preTrial = true
```

Add to any start/end condition

Add `element.start/end.and` = string that is any MATLAB expression for PsychBench to evaluate to `true`/`false` during the experiment. See reference.

Wait from any start/end condition

Add `element.start/end.timeFrom` = time (sec)

Start/End an element at multiple conditions, whichever occurs first

```
element.start/end(1).<field> = ...
element.start/end(2).<field> = ...
...
```

Triggers/Sync

Register a trigger

For any response handler element (*keyPress*, *cedrusPress*, etc.), register input as a trigger:
`element.registerTrigger = true`

Or use other element types that only record triggers, e.g. *portSender*, *portReceiver*.

Sync the experiment at a trigger

`element.syncExperiment = true`

You can then set element or trial timing from sync—see above.

Visual elements

Set screen measurements for visual angle degree units (deg)

PsychBench asks when you run an experiment.

Or set screen object properties in the experiment:
`screen.height_cm` = display panel height (cm)
`screen.distance_cm` = distance from eye to panel (cm)

Element position on screen

`element.position = [x y]` (deg), + = right/down, `[0 0]` = screen center
Default: screen center
Use `element.vary` to set movement/drift.

`element.depth` = number, + = backward
Default: `0`

Use other distance units for any property that uses deg

```
element.<property> = {value, "unit"}
"unit" = "deg", "deg-", "cm", "ww", "wh", "wwh", "px"
```

Response from subject

Record response from subject

Use a response handler element.
See element types *keyPress*, *mouseClick*, *cedrusPress*, etc.

Translate response

```
responseHandlerElement.translateResponse = [
    raw response value,  value to translate to
    raw response value,  value to translate to
    ...
    ]
```

Or a 2-column cell array if any value is not a number.
Or for custom translation use any MATLAB expression. See reference.

Score response

```
responseHandlerElement.scoreResponse = true
responseHandlerElement.correctResponse = correct response value
```

Scoring is by comparison using MATLAB *isequaln*.
Or for custom scoring use any MATLAB expression. See reference.

Feedback

Add elements (e.g. *text*, *sound*) and set them to run from response.

e.g.

```
element.start.response = true
element.start.and = "responseScore = true"
```

→ feedback at any response scored `true`

```
element.start.response = true
element.start.and = "responseScore = false"
```

→ feedback at any response scored `false`

Experiment results output

See property values for an object in results (input and record properties)

```
object.report = ["property" "property" ... ]
```

See custom information for an object in results

```
object.info.heading1 = 2;
object.info.heading2 = "Bob";
...
```

See response from subject in results

Include any of the following response handler element record properties in *report*:

response, *responseScore*, *responseTime*, *responseLatency*, *d_responseTime*, *numResponses*