Preface

Basic configuration

Overview

In this section, we will go through some basic setup of bandit task.

Conceptual Introduction

- `NUM_TRIALS` : Number of trials in the task
- `NUM_ARMS` : Number of arms in the task
- `REWARD_TYPE` : Type of reward in the task
  - "binary": Reward is either 0 or 1, following a Bernoulli distribution. Reward probability is given by the reward matrix.
  - "numerical": Reward is a integer between 0 and 100, drawn from a Gaussian distribution, whose mean is given by the reward matrix.
- `FEEDBACK_VERSION` : Version of feedback in the task
  - "full": Feedback is given for every arm whether it is chosen or not
  - "contingent": Feedback is given only for the chosen arm
- `COVER_STORY` : Cover story of the task
  - "social": Trading with aliens
  - "non-social": Mining at different ores

Setup with UI

Every field shown above can be set in UI.

> **Note**
>
> `SEED` is also in UI, but it is only for illustration purpose (To ensure the reproducibility of the reward matrix). It does not affect the actual task in any way.

Manipulation configuration

Overview

In this section, we will go through setup of three manipulations that will influence the bandit task performance, but not directly through the reward matrix.

Conceptual Introduction

Manipulation Aspects

- **Information**: (Horizontal task, Wilson et al., 2014) Information is manipulated through chosen count in the forced choice phase. Being chosen more brings more information.
- **Noise**: (Two-armed bandit task, Gershman, 2018) Noise is manipulated through the variance level of the reward generation process. Larger variance means more noise.
- **Cost**: (Search task, Bhatia et al., 2021) The cost is manipulated through the price of sampling an arm (can be regarded as a discount of reward mean). Higher price means higher cost.

> **Note**
>
> Noise and cost manipulation are only available for numerical reward type.

Manipulation Levels

There are four possible levels of each manipulation.

- "None": N = 0
- "Low": N = 1
- "Medium": N = 5
- "High": N = 10
  For Information, N is the number of forced choices; for Noise and Cost, N is the variance and price of the reward generation process.

Two Asymmetries

Those manipulations can be asymmetric between sessions, being **identical among arms** but different across sessions (e.g. Session1 has high noise than Session2). In this case, `Level` field needs to be specified. The manipulation for each arm will be set to the `Level` value.
Or they can be asymmetric within session, being **different among arms** (e.g. Arm1 has high

information than Arm2). In this case, `Level` field is not needed. The manipulation for each arm is sampled from the manipulation level list.

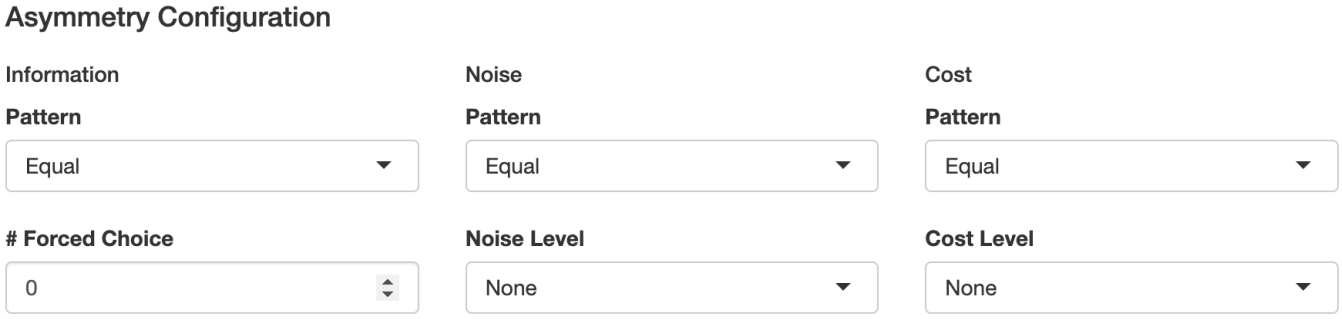
> **Note**
>
> The manipulations are always consistent within a session, across different trials.

Availability to Participants

All of those manipulations can be explicitly informed to participants. For example, by showing participants the variance of each arm, they will know the noise levels.
Or manipulations can be hidden from participants, in which case participants can only infer the levels based on their own experience.

> **Note**
>
> Information manipulation (forced choice) can only be explicit.
> Noise is usually hidden; while cost is usually explicit.

Setup with UI



- `Pattern` specifies whether the manipulation is the same or different among arms.
  - "Equal": Identical among arms, equal to `Level`.
  - "Unequal": Different among arms, sampled from the manipulation level list (0, 1, 5, 10)
- `Level` field is only available when `Pattern` = "Equal". It determines the level of the manipulation.
- `Explicitly Provided`: Only for noise and cost manipulation. Whether the manipulation is explicitly informed to participants.
- `# Forced Choice`: Only for information manipulation with `Pattern` = "Unequal". It determines the total number of forced choices in the forced choice phase. The samples will

be scaled to avoid undesired excessive forced choices. For example, if there are only 10 trials, it's easy for the sum of unequal samples to exceed the total trial number.

> **Note**
>
> Exceed the boundary (reward, trial)
> #todo

Variable configuration

Overview

In this section, we will go through the setup of the variables.
This setup will determine the sequence of the variables across trials or arms, which is crucial for the reward generation (next chapter `Reward Configuration`) and stimulus presentation.

Conceptual Introduction

Every variable needs two properties to determine its sequence across trials or arms:

- Levels: How many levels that variable has
- Pattern: How to allocate its levels across its dimension (trials/arms)
  - `Shuffle`: Randomly assign levels with guaranteed similar number of each level
  - `Loop`: Repeating sequence with ascending index
  - `Random`: (less frequently used) Random assignment with equal probability
    The resulting allocation matrix is a one-hot encoding matrix, with shape `total_length x levels`. One represents for that specific trial/arm, this variable takes this level.

Every variable belongs to one of the two categories: **trial-based** or **arm-based**.
For each category, there can be multiple variables. The allocation matrix of each variable is independent.
By default, trial-based category has one basic variable, `Trial`,indicating the trial index, with levels of `NUM_TRIALS`, following the "Loop" pattern; arm-based category has one basic variable, `Arm`, indicating the arm index, with levels of `NUM_ARMS`, following the "Loop" pattern.

> **Example**
>
> Let's say there are five trials (`NUM_TRIALS`=5), the allocation matrix of `Trial` is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This means for every trial, it has a unique ordered index, from 1 to `NUM_TRIALS`.
Similarly, there are four arms (`NUM_ARMS`=4), the allocation matrix of `Arm` is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This means for every arm, it has a unique ordered index, from 1 to `NUM_ARMS`.

In our setting, we have one trial-based variable for stimuli, `Planet`, and two arm-based variables for stimuli, `Color` and `Shape`. Those variables shall determine the stimulus presentation.

Example

`Planet` is a trial-based variable, as it might change across trials.
Let's say we want there to be three planets, and planets changes in a certain order (e.g. Trial1-Earth, Trial2-Venus, Trial3-Mercury, Trial4-Earth, Trial5-Venus).
Then `Planet` should be put in the `Trial-Based Variable` tab, with `levels` = 3, and allocation `pattern` = "Loop".
The resulting allocation matrix of `Planet` is:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Row are trials, and columns are possible planets. $M_{1,1} = 1$ means on the first trial, we are on the first planet, Earth.

`Color` is an arm-based variable, as it might be different among arms.
Let's say we want there to be two colors, and colors are randomly assigned to 4 different arms, with similar number of each color (e.g. Arm1-Orange, Arm2-Red, Arm3-Red, Arm4-Orange).
Then `Color` should be put in the `Arm-Based Variable` tab, with `levels` = 2, and

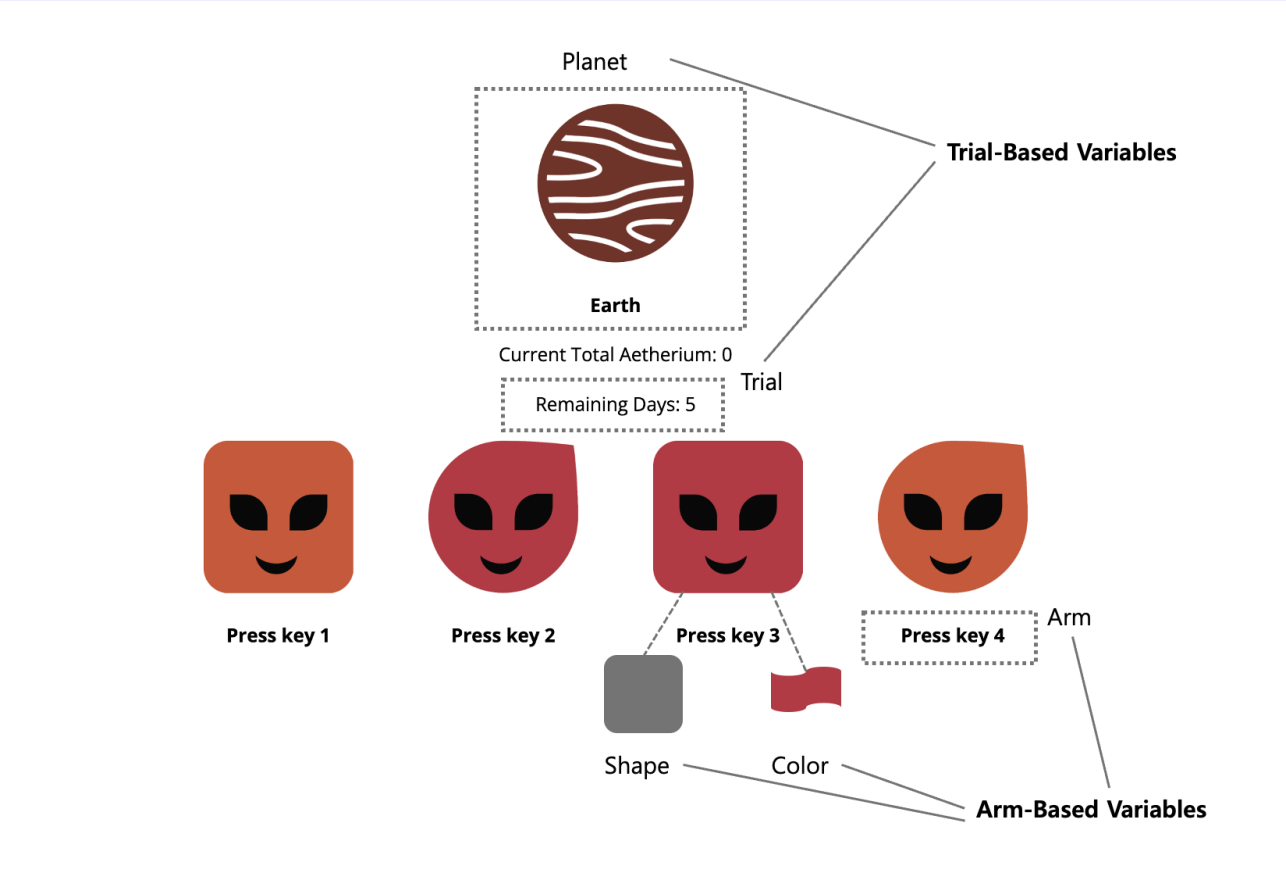
allocation `pattern` = "Shuffle".
The resulting allocation matrix of `Color` is:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Row are arms, and columns are possible colors. $M_{3,1} = 1$ means the third arm takes the first color, Red.
Similarly for arm-based variable `Shape`, the following matrix represents two shapes randomly assigned to 4 different arms (e.g. Arm1-Square, Arm2-Droplike, Arm3-Square, Arm4-Square):

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

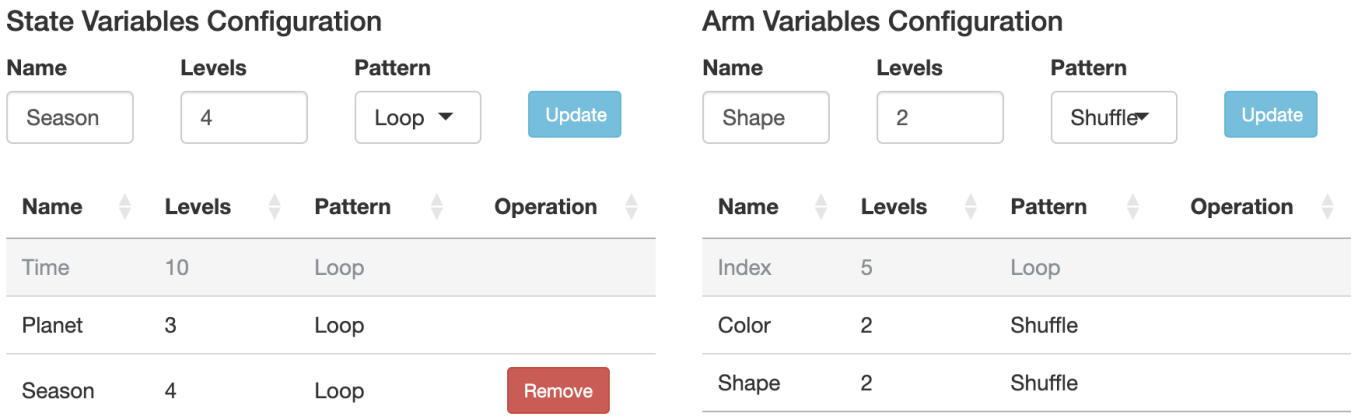
Together, our stimulus presentation on the first trial looks like this:



Besides those basic or stimulus-related variables, you can also add your own variables. For example, `season` can be a trial-based variable, with levels of `4` (e.g. Spring, Summer, Autumn, Winter), following the "Loop" pattern. By doing so, you might be able to create a

seasonality in reward generation (see `Reward Configuration` chapter).
Although those self-defined variables can influence the reward generation, they are not gonna be explicitly presented in the task, unless you modify the stimulus presentation part of the experiment script.

Setup with UI



To **add** a variable:

1. Select the corresponding panel for trial-based or arm-based variable.

   > If the variable changes across trials, go to `Trial-Based Variable`.
   > If the variable changes across arms, go to `Arm-Based Variable`.

2. Type the name of the variable in the `Name` field.
3. Set the number of levels in the `Levels` field.

   > How many different possible values that this variable might take.

4. Select the pattern of the variable in the `Pattern` field.

   > How to allocate its levels across its dimension (trials/arms).

5. Click `Update` to add the variable to the list.

To **delete** a variable, click the Delete button on the right of the variable.
To **edit** a variable, declare and update it like "add". It will automatically overwrite the existing record.

> **Note**
>
> Stimulus-related variable (`Planet`, `Color`, `Shape`) are not deletable but editable.
> Basic variables (`Trial`, `Arm`) are neither editable or deletable.

If you change the number of trials (`NUM_TRIALS`) or arms (`NUM_ARMS`) from the previous setup (`Basic Configuration`), the levels of basic variables will be automatically updated.

Reward configuration