

1 Basic configuration

1.1 Overview

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

1.2 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

1.3 Setup with UI

Number of Trials	Number of Arms	Feedback Version
<input type="text" value="10"/>	<input type="text" value="5"/>	<input type="text" value="contingent"/>
Reward Type	Cover Story	Seed (Only for UI Illustration)
<input type="text" value="numeric"/>	<input type="text" value="non-social"/>	<input type="text" value="42"/>

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.

2 Feature configuration

2.1 Overview

In this section, we will go through the setup of the features.

This setup will determine the sequence of the features across trials or arms, which is crucial for the reward generation (next chapter [Reward Configuration](#)) and stimulus presentation.

2.2 Conceptual Introduction

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

- **Levels:** How many levels that feature 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 feature takes this level.

Every feature belongs to one of the two categories: **trial-based** or **arm-based**.

For each category, there can be multiple features. The allocation matrix of each feature is independent.

By default, trial-based category has one basic feature, `Trial`, indicating the trial index, with levels of `NUM_TRIALS`, following the "Loop" pattern; arm-based category has one basic feature, `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 feature for stimuli, `Planet` , and two arm-based features for stimuli, `Color` and `Shape` . Those features shall determine the stimulus presentation.

≡ Example

`Planet` is a trial-based feature, 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 Feature` 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 feature, 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 Feature` tab, with `levels` = 2, and allocation `pattern` = "Shuffle".

The resulting allocation matrix of `Color` is:

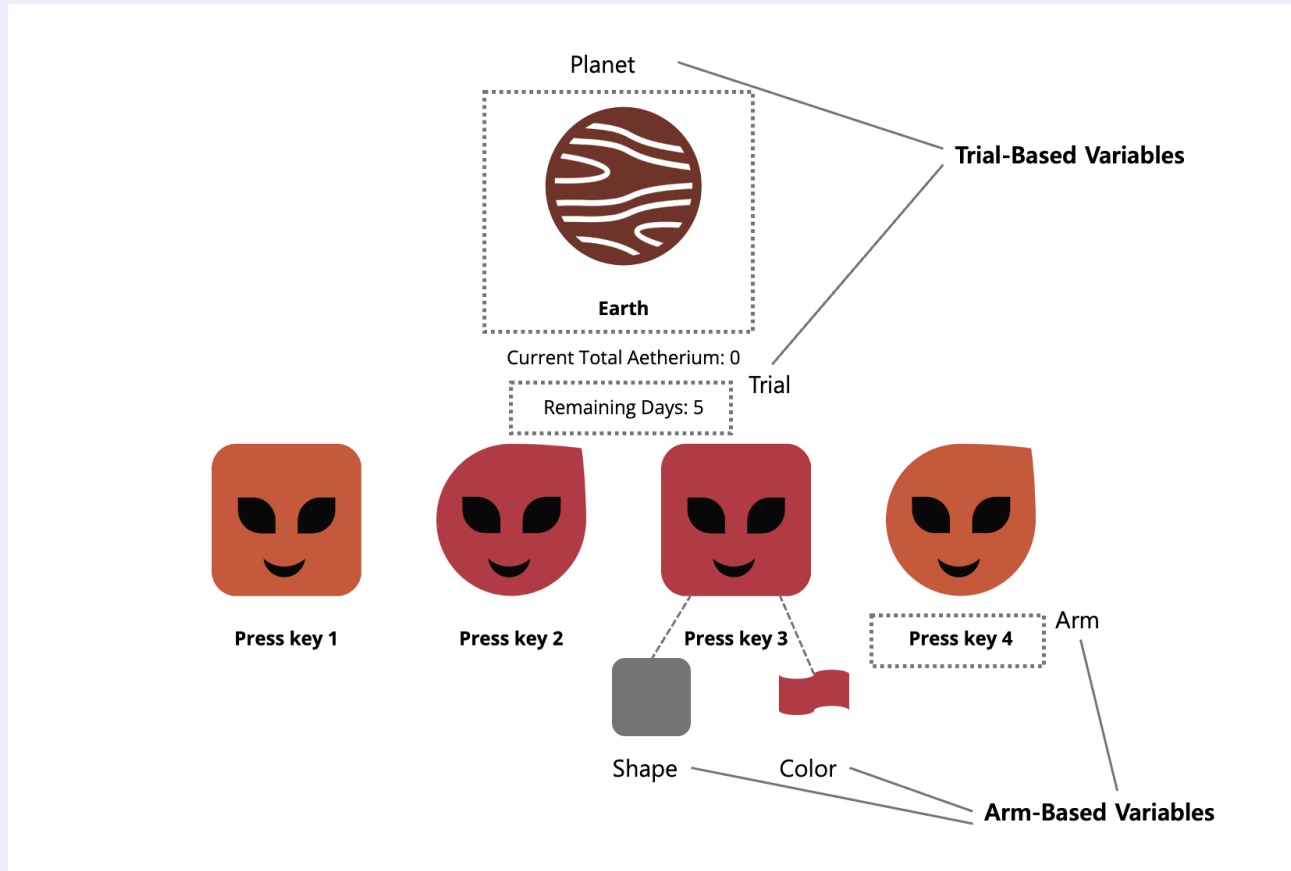
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

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 feature `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 features, you can also add your own features.

For example, `season` can be a trial-based feature, 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 features 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.

2.3 Setup with UI

State Variables Configuration

Name	Levels	Pattern	
Season	4	Loop	<button>Update</button>

Name	Levels	Pattern	Operation
Time	10	Loop	
Planet	3	Loop	
Season	4	Loop	<button>Remove</button>

Arm Variables Configuration

Name	Levels	Pattern	
Shape	2	Shuffle	<button>Update</button>

Name	Levels	Pattern	Operation
Index	5	Loop	
Color	2	Shuffle	
Shape	2	Shuffle	

To **add** a feature:

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

If the feature changes across trials, go to **Trial-Based Feature**.

If the feature changes across arms, go to **Arm-Based Feature**.

2. Type the name of the feature in the **Name** field.

3. Set the number of levels in the **Levels** field.

How many different possible values that this feature might take.

4. Select the pattern of the feature in the **Pattern** field.

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

5. Click **Update** to add the feature to the list.

To **delete** a feature, click the Delete button on the right of the feature.

To **edit** a feature, declare and update it like "add". It will automatically overwrite the existing record.

Note

Stimulus-related feature (**Planet** , **Color** , **Shape**) are not deletable but editable.

Basic features (**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 features will be automatically updated.

3 Reward configuration

3.1 Overview

The reward matrix is the core of the bandit task. Simply put, the reward matrix specifies how rewarding a arm on a given trial is.

3.2 Conceptual Introduction

3.2.1.1 Decomposition of Total Reward Matrix

The reward matrix is a matrix with shape `NUM_TRIALS x NUM_ARMS`. Each value represents for the reward of a specific arm on a specific trial.

There can be multiple trial-based features and arm-based features, see `Feature Configuration` chapter for details.

The total reward matrix is the sum of all reward matrices of every trial-based feature and arm-based feature pair.

Example

Consider the following experiment setting:

- 2 trial-based features: `Planet`, and basic `Trial` feature
- 3 arm-based features: `Color`, `Shape`, and basic `Arm` feature

The total reward matrix is the sum of the reward matrices of all trial-based feature and arm-based feature pairs:

- Reward matrix of `Trial` and `Color`
- Reward matrix of `Trial` and `Shape`
- Reward matrix of `Trial` and `Arm`
- Reward matrix of `Planet` and `Color`
- Reward matrix of `Planet` and `Shape`
- Reward matrix of `Planet` and `Arm`

3.2.1.2 Reward Matrix for a Specific Feature Pair

Then our goal is to generate the reward matrix for a specific trial-based feature and arm-based feature pair.

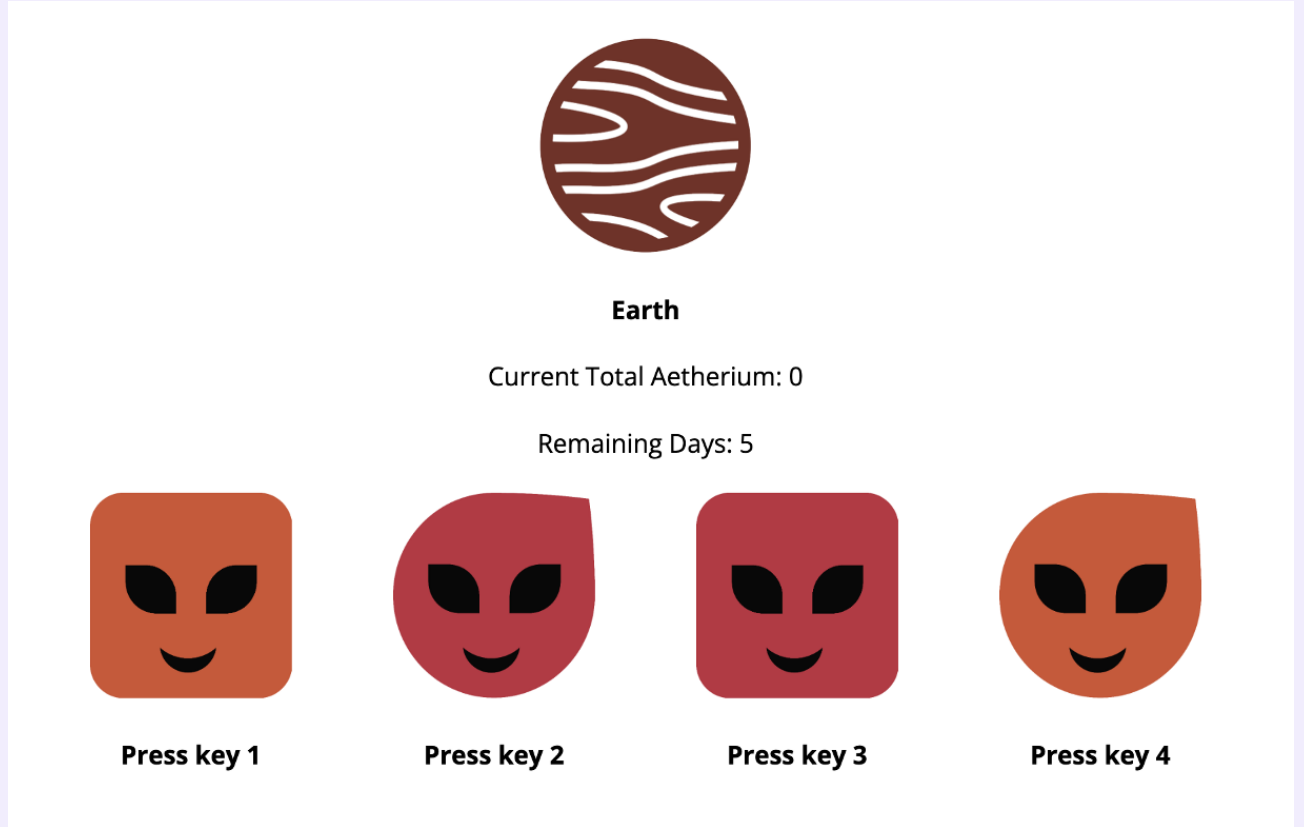
In `Feature Configuration` chapter, we've got the allocation matrix of each trial-based or arm-based feature along its dimension (trials/arms).

To generate the corresponding reward matrix, we also need the mapping matrix, which specifies the reward for each combination of the levels of trial-based feature and arm-based feature.

Then, we can multiply the trial-based allocation matrix, mapping matrix, and arm-based allocation matrix (transposed), to get the reward matrix for this specific feature pair.

Example

Consider the fourth pair in the above experiment setting: Planet and Color .



The allocation matrix of Planet $M_{Trial, Planet}$ is:

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

The allocation matrix of Color $M_{Arm, Color}$ is:

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

Let's say the mapping matrix between the levels of Planet and Color $M_{Planet, Color}$ is:

$$\begin{bmatrix} 3 & 2 \\ 1 & 2 \\ 2 & 2 \end{bmatrix}$$

Rows are levels of trial-based feature `Planet` , and columns are levels of arm-based feature `Color` . This matrix means:

- On the first Planet, Earth, the first color, Red, is assigned a reward of 3; while the second color, Orange, is assigned a reward of 2.
- On the second Planet, Venus, the first color, Red, is assigned a reward of 1; while the second color, Orange, is assigned a reward of 2.
- On the third Planet, Mercury, the first color, Red, is assigned a reward of 2; while the second color, Orange, is assigned a reward of 2.

Then, the reward matrix for this feature pair is:

$$M_{Trial,Planet} \times M_{Planet,Color} \times M_{Color,Arm} = M_{Trial,Color}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 3 & 2 \\ 1 & 2 \\ 2 & 2 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 3 & 2 \\ 2 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 3 & 3 & 2 \\ 2 & 1 & 1 & 2 \end{bmatrix}$$

3.2.1.3 Mapping Type

The mapping matrix specifies the reward for each combination of the levels of trial-based feature and arm-based feature. It can be of two major types:

- Independent mapping:
 - Trial-based: The reward changes across trials but identical among arms.
 - Arm-based: The reward changes among arms but identical across trials.
- Conditional mapping:
 - Trial-based on Arm-based: The reward of each arm changes across trials. Change pattern parameterization differs among arms.
 - Arm-based on Trial-based: The reward of each trial varies among arms. Variation pattern parameterization differs across trials.

Mathematically, the "Conditional mapping" can be regarded as repeating the "Independent mapping" for each arm with unique sampling seed.

3.2.1.4 Mapping Pattern

`Mapping Type` just tells us **where** the change happens. `Mapping Pattern` specifies **how** the change happens.

There are four available patterns:

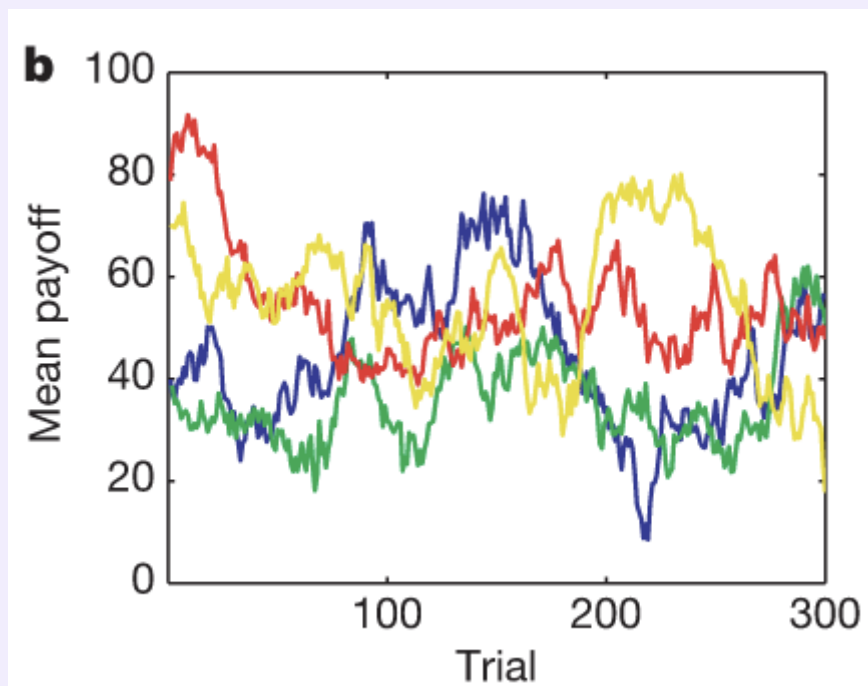
- **Identical**: The reward is identical across all levels. (No change)
- **Uniform**: The reward of each level is independently sampled from a uniform distribution.
- **Monotonic**: Two endpoints are independently sampled from a uniform distribution. The reward of each level is then linearly interpolated between the two endpoints. The resulting distribution is linearly ascending or descending.
- **Random Walk**: A sequence of rewards is generated from a random walk process. The sequence is then assigned to each level.

Example

Here are some examples of the mapping matrix for different mapping types and patterns.

Case 1: Restless Bandit (Daw et al., 2006)

"The rewards vary in a random walk manner across trials, but the patterns differ among different arms."



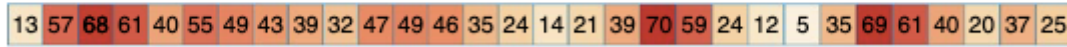
Mapping type: Conditional mapping (Trial-based [trial index] on Arm-based [arm index])

Mapping pattern: Random Walk

Case 2.1: Structural Bandit (Wu et al., 2018)

"Arms with close locations have similar rewards."

Rough



Smooth

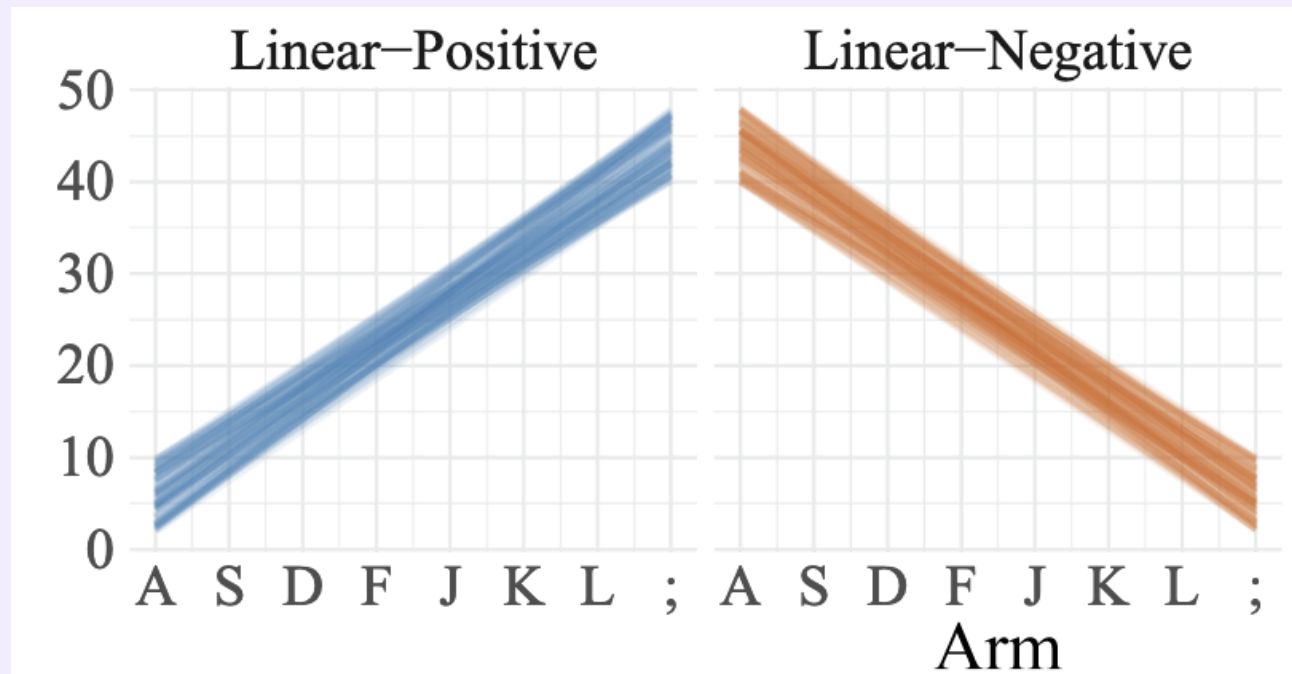


Mapping type: Independent mapping (Arm-based [arm position])

Mapping pattern: Random Walk

Case 2.2: Structural Bandit (Gershman et al., 2020)

"Arms on the right side of the screen have higher rewards."



Mapping type: Independent mapping (Arm-based [arm position])

Mapping pattern: Monotonic

Case 3: Contextual Bandit (Schulz et al., 2018)

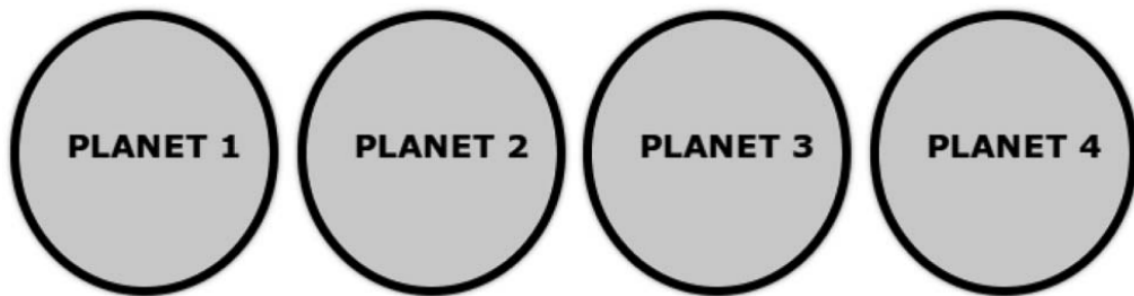
"For each arm, the reward varies across contexts, and the variation pattern is different among arms."

Number of trials left: 150
Number of Emeralds mined: 0

MERCURY: +

KRYPTON: -

NOBELIUM: +

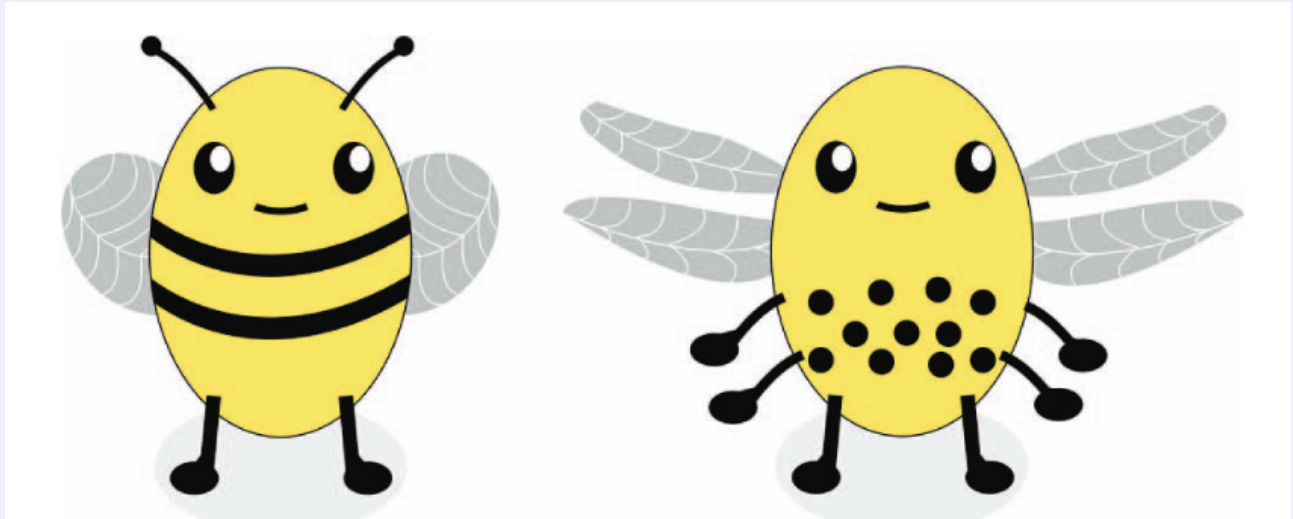


Mapping type: Conditional mapping (Arm-based [arm index] on Trial-based [context])

Mapping pattern: Uniform

Case 4: Learning Trap (Rich and Gureckis, 2018)

"For each arm, there are two dimensions of arm features. One is reward-related (# wings), and the other is unrelated (# legs)."



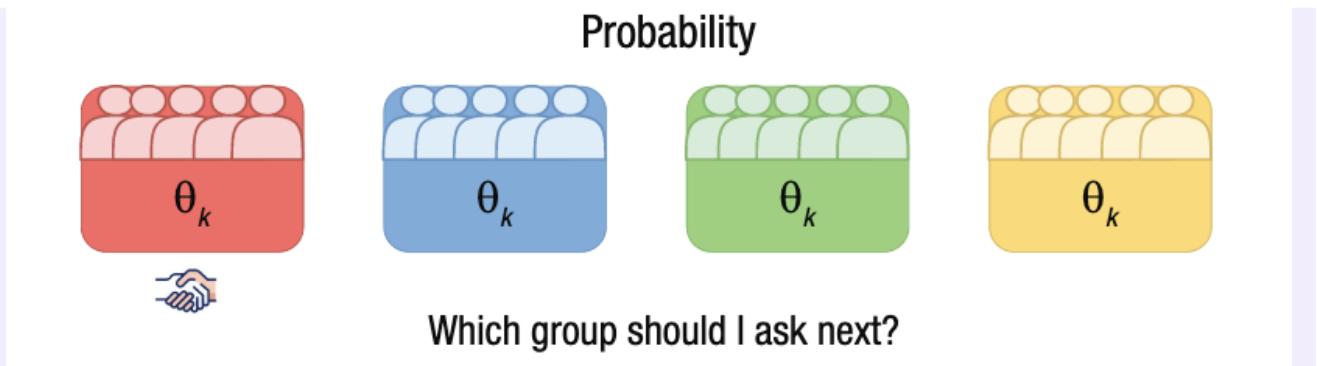
Only one of the perceptual dimensions is reward-related:

Mapping type: Independent mapping (Arm-based [wing number])

Mapping pattern: Uniform

Case 5: Multi-Armed Bandit (Bai et al., 2022)

"The reward probability of each arm is identical."



Mapping type: Independent mapping (Arm-based [arm index])

Mapping pattern: Identical

Tips

According to this idea, you can generate other interesting reward matrices. For example:

Group-based Dynamic Bandit

"The reward changes monotonically across trials, with the change pattern being identical within the same color."

Mapping type: Conditional mapping (Trial-based [trial index] on Arm-based [color])

Mapping pattern: Monotonic

You can also utilize the additivity of reward matrix to generate more complex reward matrices. For example:

Structural Bandit + Restless Bandit

"The reward varies among arms, with the variation pattern being different across planets. At the mean time, the reward of each arm changes across trials in an independent random walk manner."



Mapping type: Conditional mapping (Arm-based [arm index] on Trial-based [Planet]) &


Conditional mapping (Trial-based [trial index] on Arm-based [arm index])

Mapping pattern: Uniform & Random Walk

3.3 Setup with UI

Reward Distribution Configuration

State Variable	State Distribution		Arm Distribution	Arm Variable	
Time	Independent		Identical	Index	

State Variable	State Distribution	Interaction	Arm Distribution	Arm Variable	Operation
Time	Independent	on	Identical	Index	

Configure the mapping just like how we write regression formula:

$$f_1(\text{Feature1}|\text{Feature2}) + f_2(\text{Feature1}) + f_3(\text{Feature3}|\text{Feature4}) \dots$$

Example

- **Restless Bandit:** $f_{\text{RandomWalk}}(\text{Trial}|\text{Arm})$
For each arm, the reward changes across trials in a random walk manner. The change pattern is different among arms.
- **Structural Bandit:** $f_{\text{Monotonic}}(\text{Arm})$
Among arms, the reward changes in a monotonic manner. The change pattern is consistent across trials.
- **Contextual Bandit:** $f_{\text{Uniform}}(\text{Color}|\text{Planet})$
For each planet, the rewards of colors are drawn from a uniform distribution. The sampling pattern is different across planets.
- **Multi-Armed Bandit:** $f_{\text{Identical}}(\text{Arm})$
Among arms, the reward of each arm is identical. This pattern is consistent across trials.
- **Learning Trap:** $f_{\text{Uniform}}(\text{Wing}) + f_{\text{Identical}}(\text{Leg})$
Different wings have different rewards, drawn from a uniform distribution; while legs have identical reward. This pattern is consistent across trials.

4 Manipulation configuration

4.1 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.

4.2 Conceptual Introduction

4.2.1.1 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.

4.2.1.2 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.

4.2.1.3 Two Types of Asymmetry

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.

4.2.1.4 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.

4.3 Setup with UI

Asymmetry Configuration

Information	Noise	Cost
Pattern	Pattern	Pattern
<input type="text" value="Equal"/>	<input type="text" value="Equal"/>	<input type="text" value="Equal"/>
# Forced Choice	Noise Level	Cost Level
<input type="text" value="0"/>	<input type="text" value="None"/>	<input type="text" value="None"/>

- **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.