Guideline for rational agent framework

June 29, 2023

```
> library(dplyr)
> library(tidyr)
> library(modelr)
> knitr::opts_chunk$set(echo = TRUE)
```

This document walks through how to apply the rational agent framework to your visualization experiment. We provide general guidance and walk through the forecast visualization example in the paper in more detail.

1 Identifying the decision problem

Before we get started to applying the framework to a visualization experiment, we first need to identify the decision-making problem that forms the task in the experiment. Identify the decision problem involves identifying:

- **states** (the description of reality we want help users understand with visualization, e.g. whether the temperature is above or below freezing in our weather forecast example).
- **data generating model** (the process that generates the stimuli, i.e. the joint distribution between the state and the visualization. In the weather forecast example, the visualization is the distribution where the temperature is drawn, and the state is freezing / not freezing).
- an **action space** (the report space that the experiment requires users to use, e.g. whether or not to salt the parking lot).
- **signals** (the visualizations you show to users for helping them make decisions).
- a **scoring rule** (the quality score or payoff that the experiment assigns to participants depending on their action and the state in reality, e.g. how many points you lose if you salt when it doesn't freeze or don't salt but it freezes).

For the purpose of these guidelines, we will assume an experiment that is a slight variation on the weather forecast example in the paper, where we are asking users to report two actions. The first is the binary decision on whether to salt the parking lot and the second is their belief about the possibility of freezing in discrete levels after view a visualization about daily lowest temperature. The decision-making problem can be formalized as:

- states: $\theta \in \Theta = \{0 = \text{not freezing}, 1 = \text{freezing}\}$.
- data generating models: daily low temperature $t \sim N(\mu, \sigma^2)$; $\Pr[\theta = 1] = \Pr[t \leq 0]$; $\mu = 5$ fixed and σ uniformly from $\{2, 3, 4, 5\}$.
- action space: $a \in A = \{0 = \text{not salt}, 1 = \text{salt}\}$; $b \in B = \{0, 0.02, 0.04, 0.06, ..., 0.96, 0.98, 1\}$, reporting the belief of freezing in discrete levels.
- signals: $v \in V$ showing daily low temperature t.
- scoring rule: $S_{decision}(a, \theta)$ for reporting the decision of salting, see equation 3 in the paper; $S_{belief}(b, \theta) = S_{decision}(\arg\max_a S_{decision}(a, b), \theta)$ for reporting the belief of freezing.

The following R code defines the scoring rule $S_{decision}$ and S_{belief}. For $S_{decision}$, we follow the payoff function in equation 3 in the paper. For S_{belief}, we score belief by the optimal decision on belief, i.e. assuming the agent makes the decision rationally on their belief, what would be their score?

```
> belief_space = seq(0, 1, 0.02)
> decision_space = c(0, 1)
> decision_scoring_rule = function(decision, state) {
+   # See equation 3 in the paper
+   -100 * (1 - decision) * state - 10 * decision * (1 - state)
+ }
> belief_scoring_rule = function(belief, state) {
+   # apply the optimal decision based on the belief
+   decision = decision_space[apply(matrix(unlist(lapply(decision_space, function(d) {
+       decision_scoring_rule(d, belief)
+   })), ncol = length(belief)), 2, which.max)]
+
+   decision_scoring_rule(decision, state)
+ }
```

2 For the purposes of this walkthrough only: Simulate behavioral data using quantal response equilibrium

For the purposes of our walkthrough of the rational agent approach, we generate mock experimental data for the weather forecast example described above using quantal response equilibrium[1] (a type of behavioral model that assumes the probability of any particular action being chosen is positively related to the payoff from that action, i.e. very bad actions are unlikely), where we keep the distribution of sigma, i.e. the distribution of data generating models, as uniform distribution.

```
> p_d = c(0.25, 0.25, 0.25, 0.25)
> # p(d) the prior knowledge of the distribution of data generating models
>
> sigma_choices = c(2, 3, 4, 5)
> mu = 5
> data = data.frame(
+   mu = mu,
+   sigma = sigma_choices,
+   vis = c("mean", "mean+interval", "gradient", "HOPs")
+ ) %>%
+   data_grid(mu, sigma, vis) %>%
+   rowwise() %>%
+   mutate(
+     freezing_prob = pnorm(0, mu, sigma),
+     behavioral_decision = list(sample(decision_space, 100,
+       replace = TRUE,
+       # quantal response equilibrium
+       prob = exp(0.2 * decision_scoring_rule(decision_space, freezing_prob)))),
+     behavioral_belief = list(sample(belief_space, 100,
+       replace = TRUE,
+       # quantal response equilibrium
+       prob = exp(belief_scoring_rule(belief_space, freezing_prob))))
+   ) %>%
+   unnest(cols = c(behavioral_decision, behavioral_belief))
> head(data)
```

```
# A tibble: 6 × 6
     mu sigma vis      freezing_prob behavioral_decision behavioral_belief
  <dbl> <dbl> <chr>            <dbl>               <dbl>             <dbl>
1     5     2 gradient       0.00621                   0              0.12
2     5     2 gradient       0.00621                   0              0.34
3     5     2 gradient       0.00621                   0              0.32
4     5     2 gradient       0.00621                   1              0.2
5     5     2 gradient       0.00621                   0              0.38
6     5     2 gradient       0.00621                   0              0.14
```

[1] https://www.sciencedirect.com/science/article/pii/S0899825685710238

3 Pre-experiment analysis

In this section, we will show how to calculate the three pre-experiment quantities (rational baseline, rational benchmark, and value of information) that the rational agent framework offers. The rational agent's prior belief is before seeing the visualization, the unconditional probability of each state is drawn according to the data generating model. the rational agent will update their belief on data generating models by the implied distribution of them in visualizations. For the purpose of walking through the steps entailed in applying the framework, we provide general pseudo code first, then walk through the specific example about weather forecast with full code that assumes as behavioral data is the simulated data set generated above.

3.0.1 Rational baseline

To calculate the rational baseline, we first calculate the prior belief of rational agent and their associated action choice given these beliefs. Then we calculate what score the rational agent would get under the scoring rule if they complete the experiment with only prior knowledge.

Pseudo code See algorithm 1.

Algorithm 1: Rational baseline

```
Input: the experimental data $D$ with each row representing one
       experimental trial, the distribution of data generating models
       $p(d)$, the action space $A$, and the scoring rule $S$
Output: the rational baseline $baseline$
$belief \leftarrow E_{d \sim p(d)}(d)$;
/* the prior belief of the data generating models          */
$action \leftarrow \arg\max_{a \sim A} E_{\theta \sim belief}(S(a, \theta))$;
/* the action made on prior belief                          */
$payoff \leftarrow 0$;
for $row \in D$ do
    $dist \leftarrow$ data generating model used in $row$;
    $payoff \leftarrow payoff + E_{\theta \sim dist}(S(action, \theta))$;
end
$baseline \leftarrow payoff / \#$ of row in $D$;
/* rational baseline is the rational agent's expected payoff
   in the experiment                                        */
```

R in weather forecast example

```
> prior_belief = Reduce("+",
+   sapply(sigma_choices, function(m) {pnorm(0, mu, m)})) / length(sigma_choices)
```

```
> decision_payoffs = lapply(decision_space, function(a) {
+   decision_scoring_rule(a, prior_belief)
+ })
> prior_reporting_decision = decision_space[[which.max(decision_payoffs)]]
> prior_reporting_decision

[1] 0

> belief_payoffs = lapply(belief_space, function(b) {
+   belief_scoring_rule(b, prior_belief)
+ })
> prior_reporting_belief = belief_space[[which.max(belief_payoffs)]]
> prior_reporting_belief

[1] 0

> # simulate the action of rational agent with only prior knowledge on all experiments
> for (i in 1:nrow(data)) {
+   data[i, "prior_decision_payoff"] =
+     decision_scoring_rule(prior_reporting_decision, data[i, "freezing_prob"])
+   data[i, "prior_belief_payoff"] =
+     belief_scoring_rule(prior_reporting_belief, data[i, "freezing_prob"])
+ }
> prior_payoff = data %>%
+   dplyr::group_by(vis) %>%
+   summarise(prior_decision_payoff = mean(prior_decision_payoff),
+             prior_belief_payoff = mean(prior_belief_payoff))
> prior_payoff

# A tibble: 4 × 3
  vis           prior_decision_payoff prior_belief_payoff
  <chr>                         <dbl>               <dbl>
1 HOPs                          -7.96               -7.96
2 gradient                      -7.96               -7.96
3 mean                          -7.96               -7.96
4 mean+interval                 -7.96               -7.96
```

The result says the optimal expected payoff of the rational agent with only prior knowledge is -7.96 for both reporting decision and belief, and the optimal decision and belief are both 0, meaning that the rational agent with only prior belief would always report 0 and the rational baseline is -7.96.

3.1 Rational benchmark

Then we simulate the rational agent on the experimental data. After viewing the visualization signals, the rational agent will update the belief about states based on their belief about the distribution about data generating model. For example, the visualization signals vis provides an information about the possibility that this visualization could occur in data generating models d as $p(vis|d)$ and the prior belief about the distribution of data generating models is $p(d)$. Then the posterior belief about the distribution of data generating models could be $p(d|vis) = p(vis|d) \cdot p(d)/p(vis)$.

Pseudo code See algorithm 2.

Algorithm 2: Rational benchmark

```
Input: the experimental data $D$ with each row representing one
       experimental trial, the distribution of data generating models
       $p(d)$, the action space $A$, and the scoring rule $S$
Output: the rational benchmark $benchmark$
$payoff \leftarrow 0$;
for $row \in D$ do
    $vis \leftarrow$ visualization used in $row$;
    $p(d|vis) = p(vis|d) \cdot p(d)/p(vis)$;
    /* the posterior belief of the distribution of data
       generating models                                        */
    $belief \leftarrow E_{d \sim p(d|vis)}(d)$;
    /* the belief of the data generating models                 */
    $action \leftarrow \arg\max_{a \sim A} E_{\theta \sim belief}(S(a, \theta))$;
    /* the action made on posterior belief                      */
    $dist \leftarrow$ data generating model used in $row$;
    $payoff \leftarrow payoff + E_{\theta \sim dist}(S(action, \theta))$;
    /* the expected payoff with posterior information           */
end
$benchmark \leftarrow payoff/\#$ of row in $D$;
```

R in weather forecast example In the weather forecast example, the visualizations provide perfect information about the variance of data except "mean" visualization, so the updated distribution of data generating models could converge to a situation that only one data generating model has possibility with almost 1 and others are all almost 0. In this case, we assume that the rational agent would know which data generating model is using with certainty after viewing one visualization. Formally, assuming that vis provides perfect information about data generating model d_1, then

$$\begin{gathered} p(vis|d_1) \approx 1 \\ p(vis|d_2) \approx 0 \\ \dots \\ p(vis|d_n) \approx 0 \end{gathered}$$

so $p(d_1|vis) = p(vis|d) \cdot p(d)/p(vis) \approx 1$ when the visualization is fixed.

```
> for (i in 1:nrow(data)) {
+   # rational agent can infer the exact data generating model based on signal
+   if (data[[i, "vis"]] == "mean") {
+     posterior_belief = prior_belief
+   } else {
+     posterior_belief = data[[i, "freezing_prob"]]
+   }
+   posterior_reporting_decision = decision_space[[
+     which.max(lapply(decision_space, function(a) {
+       decision_scoring_rule(a, posterior_belief)
+     }))
+   ]]
+   posterior_reporting_belief = belief_space[[
+     which.max(lapply(belief_space, function(b) {
+       belief_scoring_rule(b, posterior_belief)
+     }))
+   ]]
+   data[[i, "posterior_decision_payoff"]] =
+     decision_scoring_rule(posterior_reporting_decision, data[[i, "freezing_prob"]])
+
+   data[[i, "posterior_belief_payoff"]] =
+     belief_scoring_rule(posterior_reporting_belief, data[[i, "freezing_prob"]])
+ }
> posterior_payoff = data %>%
+   dplyr::group_by(vis) %>%
+   summarise(posterior_decision_payoff = mean(posterior_decision_payoff),
+             posterior_belief_payoff = mean(posterior_belief_payoff))
> posterior_payoff
```

```
# A tibble: 4 × 3
  vis           posterior_decision_payoff posterior_belief_payoff
  <chr>                             <dbl>                   <dbl>
1 HOPs                              -5.69                   -5.69
2 gradient                          -5.69                   -5.69
3 mean                              -7.96                   -7.96
4 mean+interval                     -5.69                   -5.69
```

When seeing "mean" visualization, the rational agent didn't get the information about sigma, so they can't update the belief on data generating models, and the rational benchmark for both reporting action and belief turns out to be -7.96, which is the same as rational baseline. For the other visualization, the rational benchmark for both reporting action and belief is -5.69.

3.2 Value of information

We define the value of information in rational agent framework as the difference between rational benchmark and baseline, i.e. the maximum improvement can be made when agent has the right prior knowledge and follows the Bayesian rule to update the belief.

In the example of weather forecast, the value of information is `posterior_payoff - prior_payoff = 2.27` expect `0` for "mean" visualization.

4 Post-experiment analysis

In this section, we will calculate the score of behavioral agents and the score of the calibrated behavioral.

4.1 Behavioral score

We use the scoring rule on behavioral agents' actions.

Pseudo code See algorithm 3.

```
Algorithm 3: Behavioral score
  Input: the experimental data $D$ with each row representing one
         experimental trial, and the scoring rule $S$
  Output: the behavioral scores $behavioral$
  $payoff \leftarrow 0$;
  for $row \in D$ do
      $dist \leftarrow$ data generating model used in $row$;
      $action \leftarrow$ action made by the behavioral agent in $row$;
      $payoff \leftarrow payoff + E_{\theta \sim dist}(S(action, \theta))$;
      /* the expected score of behavioral action                */
  end
  $behavioral \leftarrow payoff/\#$ of row in $D$;
```

R in weather forecast example

```
> for (i in 1:nrow(data)) {
+   data[i, "behavioral_decision_payoff"] =
+     decision_scoring_rule(data[i, "behavioral_decision"], data[i, "freezing_prob"])
+   data[i, "behavioral_belief_payoff"] =
+     belief_scoring_rule(data[i, "behavioral_belief"], data[i, "freezing_prob"])
+ }
> behavioral_payoff = data %>%
+   dplyr::group_by(vis) %>%
+   summarise(behavioral_decision_payoff = mean(behavioral_decision_payoff),
+             behavioral_belief_payoff = mean(behavioral_belief_payoff))
> behavioral_payoff

# A tibble: 4 × 3
  vis           behavioral_decision_payoff behavioral_belief_payoff
  <chr>                              <dbl>                    <dbl>
1 HOPs                               -6.84                    -8.49
2 gradient                           -6.89                    -8.65
3 mean                               -7.02                    -8.49
4 mean+interval                      -6.97                    -8.41
```

Since the behavioral agent's actions are generated randomly but not from real persons, some of the payoffs are below rational baseline, e.g. the behavioral payoff when reporting belief. Then we will introduce the calibrated behavioral score, which can represent the behavioral in the baseline-and-benchmark scale.

4.2 Calibrated behavioral score

The rational baseline and the rational benchmark could be considered as the evaluating scale for behavioral agents when assuming that the agent gets the right prior knowledge, updates the belief by Bayesian rule, and optimizes the decision based on belief. However, unfortunately, behavioral agents may not follow these requirements, which makes them less comparable to the scale. Thus, we propose the calibrated behavioral score, where the rational agent calibrates the behavioral agent's behaviors into Bayesian and thus falls into the interval between rational baseline and benchmark.

In the calibration, the rational agent is not using the visualizations as signals but using the actions made by behavioral agents as signals. When getting a behavioral agent's action a, they will look at what's the possibilities that this action will appear in different data generating models ($p(a|d)$) and use these possibilities to update the prior belief ($p(d|a) = p(a|d) \cdot p(d)/p(a)$).

The pseudo code is as following...

Pseudo code See algorithm 4.

R in weather forecast example

```
> # the joint distribution of actions and data generating models
> joint_decision_distribution = data %>%
+   dplyr::group_by(mu, sigma, freezing_prob, behavioral_decision) %>%
+   summarise(count = n()) %>%
+   dplyr::group_by(mu, sigma, freezing_prob) %>%
+   mutate(count = count / sum(count))
> joint_belief_distribution = data %>%
+   dplyr::group_by(mu, sigma, freezing_prob, behavioral_belief) %>%
+   summarise(count = n()) %>%
```

Algorithm 4: Calibrated behavioral score

```
**Input:** the experimental data $D$ with each row representing one experimental trial, the distribution of data generating models $p(d)$, the action space $A$, and the scoring rule $S$
**Output:** the calibrated behavioral score $benchmark$
$payoff \leftarrow 0$;
for $row \in D$ do
    $signal \leftarrow$ action made by the behavioral agent in $row$
     $p(d|signal) = p(signal|d) \cdot p(d)/p(signal)$;
    /* the posterior belief of the distribution of data
       generating models in calibration                         */
    $belief \leftarrow E_{d \sim p(d|signal)}(d)$;
    /* the belief of the data generating models                 */
    $action \leftarrow \arg\max_{a \sim A} E_{\theta \sim belief}(S(a, \theta))$;
    /* the action made on posterior belief                      */
    $dist \leftarrow$ data generating model used in $row$;
    $payoff \leftarrow payoff + E_{\theta \sim dist}(S(action, \theta))$;
    /* the expected payoff with posterior information           */
end
$benchmark \leftarrow payoff/\#$ of row in $D$;
```

```
+   dplyr::group_by(mu, sigma, freezing_prob) %>%
+   mutate(count = count / sum(count))
> sum_payoff = 0
> for (i in 1:nrow(data)) {
+   decision_signal = data[[i, "behavioral_decision"]]
+
+   joint_decision_distribution_filtered = joint_decision_distribution %>%
+     filter(behavioral_decision == decision_signal)
+
+   distributions = pnorm(0, mu, sigma_choices)
+
+   p_a = nrow(data %>% filter(behavioral_decision == decision_signal)) / nrow(data)
+
+   sigmas = joint_decision_distribution_filtered$sigma
+   p_a_d = as.list(rep(0, length(sigma_choices)))
+   names(p_a_d) = as.character(sigma_choices)
+   p_a_d[as.character(sigmas)] = joint_decision_distribution_filtered$count  # p(a|d)
+   p_a_d = unname(unlist(p_a_d))
+   p_d_a = p_a_d * p_d / p_a  # p(a|d) * p(d) / p(a)
+
+   calibrated_belief = sum(distributions * p_d_a)  # E_p_d_a(d)
+   calibrated_reporting_decision = decision_space[[which.max(
+     lapply(decision_space, function(a) {
```

```
+       decision_scoring_rule(a, calibrated_belief)
+     }))]]
+   data[[i, "calibrated_decision_payoff"]] =
+     decision_scoring_rule(calibrated_reporting_decision, data[[i, "freezing_prob"]])
+
+
+   belief_signal = data[[i, "behavioral_belief"]]
+
+   joint_belief_distribution_filtered = joint_belief_distribution %>%
+     filter(behavioral_belief == belief_signal)
+
+   p_a = nrow(data %>% filter(behavioral_belief == belief_signal)) / nrow(data)
+
+   sigmas = joint_belief_distribution_filtered$sigma
+   p_a_d = as.list(rep(0, length(sigma_choices)))
+   names(p_a_d) = as.character(sigma_choices)
+   p_a_d[as.character(sigmas)] = joint_belief_distribution_filtered$count  # p(a|d)
+   p_a_d = unname(unlist(p_a_d))
+   p_d_a = p_a_d * p_d / p_a  # p(a|d) * p(d) / p(a)
+
+   calibrated_belief = sum(distributions * p_d_a)  # E_p_d_a(d)
+   calibrated_reporting_belief = belief_space[[which.max(lapply(belief_space, function(a)
+     belief_scoring_rule(a, calibrated_belief)
+   }))]]
+   data[[i, "calibrated_belief_payoff"]] =
+     belief_scoring_rule(calibrated_reporting_belief, data[[i, "freezing_prob"]])
+ }
> calibrated_payoff = data %>%
+   dplyr::group_by(vis) %>%
+   summarise(calibrated_decision_payoff = mean(calibrated_decision_payoff),
+             calibrated_belief_payoff = mean(calibrated_belief_payoff))
> calibrated_payoff

# A tibble: 4 × 3
  vis           calibrated_decision_payoff calibrated_belief_payoff
  <chr>                              <dbl>                    <dbl>
1 HOPs                               -6.84                    -5.76
2 gradient                           -6.89                    -5.77
3 mean                               -7.02                    -5.75
4 mean+interval                      -6.97                    -5.77
```

Now we get the calibrated behavioral scores. The calibrated scores of all visualization types are between rational baseline and rational benchmark.