Guidance on building a shiny app to solve POMDPs

This supplementary information provides guidance on how to build shiny apps to solve a POMDP. Here, we build on an existing SARSOP R wrapper (https://github.com/boettiger-lab/sarsop/) to call SARSOP from R (Boettiger *et al.* 2018). Interested readers can refer to the shiny website for tutorials and examples (https://shiny.rstudio.com/tutorial/).

Structure of a Shiny App

Shiny apps are contained in a single script called app.R that has three components (Figure S1):

- a **user interface object** that controls the layout and appearance of your app;
- a **server function** that contains the code to build an app;
- and a call to the **shinyApp** function that creates Shiny app objects from an explicit UI/server pair.

We briefly present the user interface object and the server function.

User interface object: shiny dashboard

In simple terms, the user interface object collects input data and displays output data.

The user interface object that we chose for our app is a shiny dashboard (https://rstudio.github.io/shinydashboard/index.html). It is also possible to use the classical ui object from shiny. In its most basic form, the shiny dashboard has the following code and appearance (Figure S 1).

```
## app.R ##
library(shiny)
library(shinydashboard)

ui <- dashboardPage(
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody()
)

server <- function(input, output) { }

shinyApp(ui, server)
```

Figure S 1: Basic code and appearance of a shiny dashboard

The dashboard is divided in three parts header, sidebar, and body. The content of these parts can be customised (see https://rstudio.github.io/shinydashboard/structure.html).

There are many challenges associated with designing an effective UI – we invite readers to refer to numerous guidance available on the topic (Kimball and Harslem, 1982; Myers and Rosson, 1992). Human computer interaction is a research domain that studies the subtle

interplay between people and computers (Shneiderman et al., 2016). In our case, we have improved our design based on end-users' feedback.

Server function

In our app, the server function executes the following tasks:

1. pre-processing the inputs by using reactive expressions. Reactive expressions are R expressions that uses the value of an/several inputs and returns a value. The reactive expression updates its value only when the original input(s) change;
2. Solving the POMDP, computing an optimal policy, and running simulations (using functions from smsPOMDP package and from the sarsop package, see SARSOP functions below);
3. Creating interactive plotly graphs (https://plotly.com/r/):
    a. One for the user to input past actions and displays optimal policy;
    b. One for the user to input past observations, displays the belief state of the species over time, and the results of simulations;
    c. One that displays the expected rewards over time obtained with the actions and observations set by the user, the rewards expected from the initial belief state under optimal policy and the rewards expected from the current belief state under optimal policy.

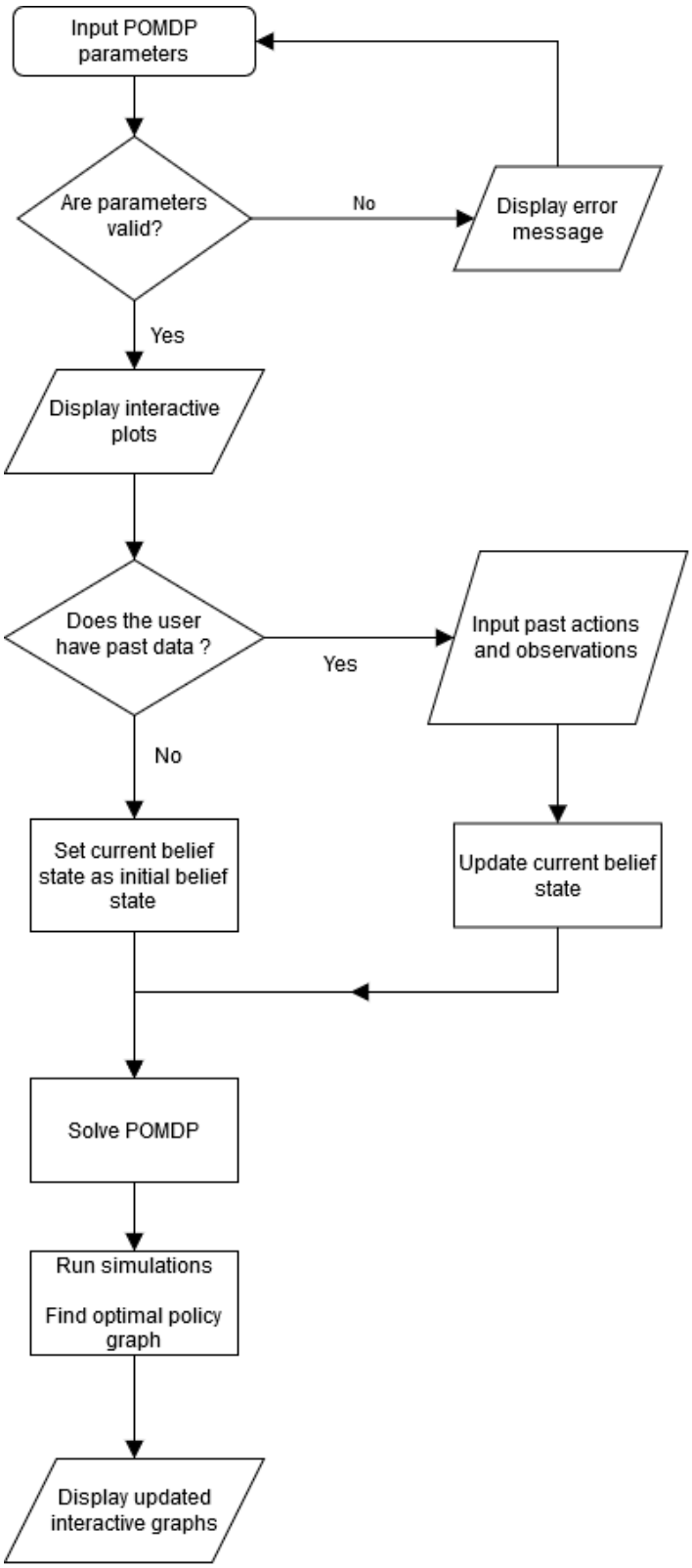
To solve a POMDP, the SARSOP package contains the following functions:

R function	Description
write_pomdpx	Writes a pomdpx file using transition, observation and reward matrices, state prior, discount factor.
pomdpsol	The pomdpsol function solves a model file (pomdpx file) and returns the path to the output policy file.
read_policyx	Read a .policyx file created by SARSOP and return alpha vectors and associated actions.
sarsop	sarsop wraps the tasks of writing the pomdpx file defining the problem, running the pomdsol (SARSOP) algorithm in C++, and then reading the resulting policy file back into R. The returned alpha vectors and alpha_action information is then transformed into a more generic, user-friendly representation as a matrix whose columns correspond to actions and rows to states. This function can thus be used at the heart of most POMDP applications.

Deploying a shiny app on the web

In some cases, deploying shiny app on the web can improve the user experience. Interested readers can refer to tutorials on https://www.shinyapps.io/.

Flowchart of the app



References

Boettiger, C, Ooms, J, Memarzadeh, M (2018) 'sarsop: Approximate POMDP Planning Software in R.' https://github.com/boettiger-lab/sarsop)

Myers, B.A. and Rosson, M.B., 1992, June. Survey on user interface programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 195-202).

Kimball, R. and Harslem, B.V.E., 1982. Designing the Star user interface. *Byte*, *7*(1982), pp.242-282.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N. and Diakopoulos, N., 2016. Designing the user interface: strategies for effective human-computer interaction. Pearson.