Distracted by Shiny Objects

Christopher R. Genovese

Department of Statistics & Data Science

10 Sep 2024
Session #5

Plan

Migit Discussion

Plan

Migit Discussion

Object Lessons

Plan

Migit Discussion

Object Lessons

Activity: State Machines

Announcements

- Email subject: [750]
- Office Hours Wed 4:30-5:30, plus others.
- Documents (TL, slides, etc.) pushed to documents repo
- **Reading**:
  - Thursday: TBA
- `migit` assignment updated, you should continue work on that. Sub-Tasks 1–8 due in Thu 19 Sep.
- **Homework**: migit Part 1 (sub-tasks 1–8) due Thursday 19 Sep.

Plan

Migit Discussion

Object Lessons

Activity: State Machines

Plan

Migit Discussion

Object Lessons

Activity: State Machines

What is Happening Here?

```
fit <- lm(dist ~ speed, data=cars)
predict(fit)
summary(fit)
```

What about?

```
path = Path(start_path)
if not path.exists():
    return None
path = path.resolve()
```

Programming with *Nouns*

An **object** is a collection of *data* together with *methods* that operate on that data.

A **class** is a Type that describes a specific set of objects.

Object-oriented programming (OOP) uses objects/classes as the central abstraction. "Objects as nouns."

Key Principles:

- Encapsulation
- Polymorphism
- Extensibility
- Inheritance (use carefully)
- Separation of Concerns
- Dependency Inversion (dependence on the abstraction not the details)
- Safety

Aside: The Many Faces of OOP in R

R has many different systems for object-oriented programming. The main ones are:

- **S3**

  The oldest and simplest system, built on lists (usually). An object is just a variable that's been labeled as having a certain class. Generic functions can be written to operate on different classes. Commonly used in base R.

- **S4**

  A more sophisticated system with inheritance, multiple dispatch, and more formality. Heavily used in some circles (e.g., Matrix, Bioconductor), and it has some big advantages. But it is less commonly used overall.

- **R6**

  A lighter weight and higher performance version of RC (Reference Classes). Has reference semantics, public/private methods, properties (called /active bindings/), and other features familiar in other languages. (See r6.r-lib.org for details.)

- **S7**

  A new OOP style that tries to combine the best of S3 + S4 (thus S7) with a more modern, ergonomic feel. (See here for details.) This is not yet in base R but is available from CRAN.

Interactive Examples

1. Images
2. Documents
3. Books and Libraries
4. Paths
5. Hyperreal Numbers
6. Monoids

Plan

Migit Discussion

Object Lessons

Activity: State Machines

State Machines

A **state machine** is an abstraction that represents a collection of possible states, allowed transitions between particlar states, actions performed at various points during transitions, and events that can be dispatched to initiate changes/actions.

Imagine a *Domain Specific Language* (DSL) for specifying a state machine:

```
state a
state b

transition one from c to a
transition two from a, b to c
transition three from b to c, d
transition six from b to d or from d to e

delegate input to one

action on enter a
  | ... arbitrary lines
  | ...
end

action during two with event, source, target
  | ...
  | ...
end
```

State Machines (cont'd)

We list possible states and name specific allowed transitions. Actions can occur

1. On entering a state (action gets state and event)
2. During a transition (action gets source and target and event)
3. On exiting a state (action gets state and event)

We can also delegate events from an arbitrary name to a transition.

The task is to define an class structure that can support this general mechanism.

You can assume that the specification is parsed for you.

Demo

The End