

Object-Oriented Programming and Lexical Scope: Exercises

1. In Basic R Programming Exercise 1, you wrote a function to fit binomial logistic-regression models by maximum likelihood employing iterated weighted least squares. Using the example in Section 8.7.1 of “Writing R Programs” (Fox and Weisberg, draft) as a model, rewrite your program to produce S3 objects of class "logit". Provide appropriate method functions for this class – at least `summary()` and `print()` methods.
2. In Basic R Programming Exercise 2, you wrote a function to fit the ordered logit and probit models. Repeat Exercise 1 for this model, creating objects of class "ordreg".
3. Taking advantage of lexical scoping, write a function named `account` that returns a function that simulates a bank account, behaving in the following manner:

```
> John <- account(100)
> John()
[1] 100
> John(deposit=50)
[1] 150
> John(withdrawal=25)
[1] 125

> Jane <- account(500)
> Jane()
[1] 500
> Jane(withdrawal=100)
[1] 400

> John()
[1] 125
```

Your `account` function should take a single argument (named, say, `balance`) that specifies the initial balance of the account, and should return a function with two arguments, `deposit` and `withdrawal`, both of which default to 0. A function that is returned by `account` (e.g., `John` or `Jane` in the example) should always return the current value of `balance`, even when it is called with no arguments; `balance` should be updated whenever the returned function is called with a nonzero `deposit` or `withdrawal` argument.

Hints: `balance` is bound in the frame of the call to `account()` and hence will be in the environment of the function that's returned. To change the value of `balance`, you can make use of `<<-`, the non-local assignment operator, which can make an assignment to `balance` in the frame of the call to `account()`. See `?`<<-``.