Week 2: Symbol Binding/Optimization/Coding Standards

R Programming: Week 2

Symbol Binding

Symbols can have global environments binding values. It's best to use the search() to make sure you aren't reusing known values.

Example 1: Lexical Scoping

```
f <- function(x,y) { x^2 + y / z}
```

- z is a free variable. Scoping rules determine how we define z.
- A value of z is searched for in the environment in which the functions are defined.
- Nice for global variables.

Example 2: Lexical Scoping

```
make.power <- function(n) {
  pow <- function(x) {
    x^n
  }
    pow
}

cube <- make.power(3)
square <- make.power(2)
cube(3)

## [1] 27

square(3)

## [1] 9
```

- n is a free variable

Example 3: Function's Environment

```
ls(environment(cube))

## [1] "n"   "pow"

get("n", environment(cube))

## [1] 3
```

Example 4: Dynamic Scoping

```
y <- 10
g <- function(x) {
    x+y
}
f <- function(x) {
    y <- 2
    y^2 + g(x)
}

f(3)

## [1] 17
```

Optimization

Optimization routines like optim, nlm, optimize require you to pass a function whose argument is a vector of paramenters (e.g. log likelihood).

Example 1: Constructor Function - Maximize normal likelihood

```
make.NegLogLik <-
        function(data,fixed=c(FALSE,FALSE)) {
   params <- fixed
  function(p) {
  params[!fixed] <- p
  mu <- params[1]
  sigma <- params[2]
  a <- -0.5*length(data)*log(2*pi*sigma^2)
  b <- -0.5*sum((data-mu)^2) / (sigma^2)
  -(a + b)
  }
}
```

Example 2: Usage of Example 1

```
set.seed(1); normals <- rnorm(100, 1, 2)
nLL <- make.NegLogLik(normals)
nLL

## function(p) {
##   params[!fixed] <- p
##   mu <- params[1]
##   sigma <- params[2]
##   a <- -0.5*length(data)*log(2*pi*sigma^2)
##   b <- -0.5*sum((data-mu)^2) / (sigma^2)
##   -(a + b)
##   }
## <bytecode: 0x7f9a17545e68>
## <environment: 0x7f9a134034a0>
```

```
function(p) {
 params[!fixed] <- p
 mu <- params[1]
 sigma <- params[2]
 a <- -0.5*length(data)*log(2*pi*sigma^2)
 b <- -0.5*sum((data-mu)^2) / (sigma^2)
 -(a + b)
 }
```

```
## function(p) {
##  params[!fixed] <- p
##  mu <- params[1]
##  sigma <- params[2]
##  a <- -0.5*length(data)*log(2*pi*sigma^2)
##  b <- -0.5*sum((data-mu)^2) / (sigma^2)
##  -(a + b)
##  }
```

```
ls(environment(nLL))
```

```
## [1] "data"   "fixed"  "params"
```

Refer to prof notes for examples optimize functions usages.

Coding Standards

- Always use text files/text editors
- Indent code
- limit the width of code to 80 columns
- Indent (4 spaces min/ 8 space ideal)
- Limit the length of individual functions