Solutions to Selected Exercises for Braun and Murdoch's A First Course in Statistical Programming with R

Kristy Alexander, Yiwen Diao, Qiang Fu, and Yu Han
W. John Braun and Duncan J. Murdoch

November 21, 2009

Chapter 7

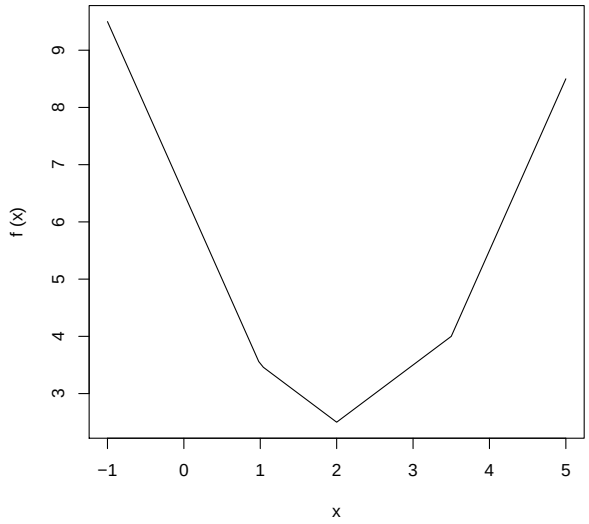
Numerical Optimization

7.1 The Golden Section Search Method

Write golden search function

```
> golden <- function(f, a, b, tol = 0.0000001){
+    ratio <- 2/(sqrt(5) + 1)
+    x1 <- b - ratio*(b - a)
+    x2 <- a + ratio*(b - a)
+
+    f1 <- f(x1)
+    f2 <- f(x2)
+
+    while( abs(b -a) > tol){
+            if(f2 > f1){
+                b <- x2
+                x2 <- x1
+                f2 <- f1
+                x1 <- b - ratio*(b - a)
+                f1 <- f(x1)
+            }else{
+                a <- x1
+                x1 <- x2
+                f1 <- f2
+                x2 <- a + ratio*(b - a)
+                f2 <- f(x2)
+            }
+    }
+    return((a + b) / 2)
+ }
```

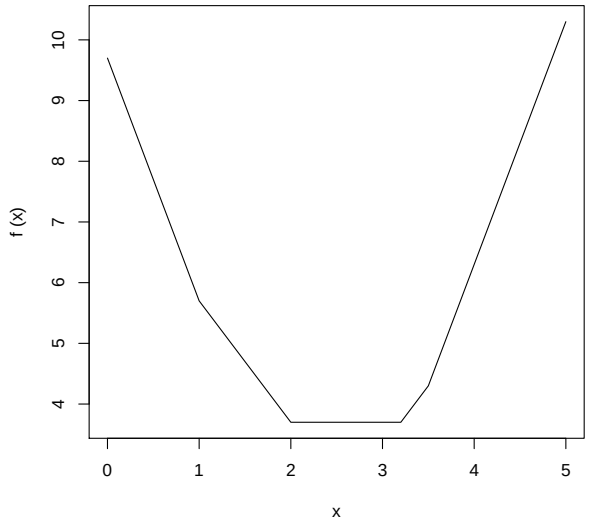
1. (a)
```
> f <- function(x) {
+     abs(x - 3.5) + abs(x - 2) + abs(x - 1)
+ }
> curve ( f, from = -1, to = 5)
```



This indicates that there is only one minimum close to zero and it is located to the right of zero.

```
> golden( f, 1, 5)
[1] 2
```

(b)
```
> f <- function(x) {
+       abs(x - 3.2) + abs(x - 3.5) + abs(x - 2) + abs(x - 1)
+ }
> curve(f, from = 0, to = 5)
```



This shows that between 1 and 4 there is more than one minimum point.

```
> golden( f, 1, 2.15)
[1] 2.15
> golden( f, 1, 5)
[1] 3.2
> golden( f, 1, 3)
```

```
[1] 3
> golden( f, 2.5, 3.5)
[1] 3.2
```

3.
```
> golden <- function( f, a, b, tot = 0.0000001){
+    ratio <- 2/(sqrt(5) + 1)
+    x1 <- b - ratio*(b - a)
+    x2 <- a + ratio*(b - a)
+
+    f1 <- f(x1)
+    f2 <- f(x2)
+
+    while( abs(b -a) > tot){
+            if(f2 < f1){
+                b <- x2
+                x2 <- x1
+                f2 <- f1
+                x1 <- b - ratio*(b - a)
+                f1 <- f(x1)
+            }else{
+                a <- x1
+                x1 <- x2
+                f1 <- f2
+                x2 <- a + ratio*(b - a)
+                f2 <- f(x2)
+            }
+    }
+    return((a + b)/2)
+ }
```

7.4 Built-in Functions

1. (a)
```
> f <- function(x) {
+      abs(x - 3.5) + abs(x - 2) + abs(x - 1)
+ }
> ans <- optimize( f, c(1, 3))
> ans$minimum
[1] 2
```
This confirms the minimum as 2.

(b)
```
> f <- function(x) {
+     abs(x - 3.2) + abs(x - 3.5) + abs(x - 2) + abs(x - 1)
+ }
> ans <- optimize( f, c(1, 4))
> ans$minimum
[1] 2.145956
```
This confirms that a minimum is 2.658372.

7.5 Linear Programming

7.5.7 Quadratic Programming

1. (a)
```
> library(lpSolve)
> a.lp <- lp(objective.in = c(1, 3, 4, 1),
```

```
+            const.mat = matrix(c(1, -2, 0, 0,
+                                 0, 3, 1, 0,
+                                 0, 1, 0, 1), nrow = 3, byrow = TRUE),
+            const.rhs = c(9, 9, 10),
+            const.dir = c(">=", ">=", ">="))
> a.lp
Success: the objective function is 31
> a.lp$solution
[1] 15  3  0  7
```

(b) The solution will not change since the optimal solution requires that all decision variables are integers.

(c)
```
> c.lp <- lp(objective.in = c(1, -3, 4, 1),
+            const.mat = matrix(c(1, -2, 0, 0,
+                                 0, 3, 1, 0,
+                                 0, 1, 0, 1), nrow = 3, byrow = TRUE),
+            const.rhs = c(9, 9, 10),
+            const.dir = c(">=", ">=", ">="))
> c.lp
Error: status 3
```

This implies that the solution is unbounded.

7.6 Chapter Exercises

1.
```
> x <- c( 0.45, 0.08, -1.08, 0.92, 1.65, 0.53, 0.52, -2.15, -2.2, -0.32, -1.87, -0.16,
+  -0.19, -0.98, -0.2, 0.67, 0.08, 0.38, 0.76, -0.78)
> y <- c( 1.26, 0.58, -1, 1.07, 1.28, -0.33, 0.68, -2.22, -1.82, -1.17, -1.54, 0.35,
+  -0.23, -1.53, 0.16, 0.91, 0.22, 0.44, 0.98, -0.98)
> library(quadprog)
> X <- cbind( rep(1, 20), x)
> XX <- t(X) %*% X
> Xy <- t(X) %*% y
> A <- matrix( c(1, -1), ncol = 1)
> b <- 0
> ans <- solve.QP( Dmat = XX, dvec = Xy, Amat = A, bvec = b)
> ans$solution
[1] 0.5314036 0.5314036
```

This implies that the intercept = slope = 0.5314036

3.
```
> A <- matrix( c( 1, 1, -1, 0, 0, 0, 0, 1, 0, 0, 1, -1, 0, 0, 1, 0, 0, 0, 0, 1, -1),
+  nrow=3, byrow=TRUE)
> D <- matrix( c( 0.01, 0.002, 0.002,
+                 0.002, 0.01, 0.002,
+                 0.002, 0.002, 0.01), nrow = 3)
> x <- c( 0.002, 0.005, 0.01)
> b <- c( 1, 0, -0.5, 0, -0.5, 0, -0.5)
> ans <- solve.QP(2*D, x, A, b, meq = 1)
> ans$solution
[1] 0.15625 0.34375 0.50000
```

5.
```
> D <- matrix( c( 0.01, 0,
+                 0, 0.04), nrow = 2, byrow = TRUE)
```

```
> A <- matrix( c( 1, 1, 1,
+                 0, 0, 1), nrow = 2, byrow = TRUE)
> x <- c( 0.005, 0.01)
> b <- c( 1, 0, 0)
```

(a)
```
> ans <- solve.QP(D, x, A, b, meq = 1)
> ans$solution
[1] 1.00 0.25
```

(b)
```
> ans <- solve.QP(2*D, x, A, b, meq = 1)
> ans$solution
[1] 1.000 0.125
```