Lecture 9

Doing Math and Simulations in R

GEOG 489

SPRING 2020

Assignment 2

Your goal is to write a local-window smoothing function for use on matrices.

The inputs are as follows:

myMatrix: an arbitrarily large numeric matrix that is at least 3 x 3

smoothingMatrix: an arbitrary 3 x 3 numeric matrix

The output should be a matrix of the same size as myMatrix. For each location in myMatrix, the value should be equal to the average value of the 3x3 window around that location multiplied element-wise by the smoothingMatrix.

Assignment 2

For example:

```
myMatrix <- matrix(1:25,nrow=5)
myMatrix
smoothingMatrix <- matrix(0.5,nrow=3,ncol=3)
smoothingMatrix
# The value of the output matrix at position 2,3 will be
# based on the local window around that point:
localWindow23 <- myMatrix[1:3,2:4]
localWindow23
# Multiplying this local window by the example smoothing matrix and
#    determining the mean value results in a single value of 6.
```

Assignment 2 is due the coming Tuesday, Feb. 25 2019 at midnight.

Math and Simulation

```
### Math Functions
# R has any basic math function can you think of:
?exp()          # Exponential function, base e
?log()          # Natural logarithm
?log10()        # Logarithm base 10
?sqrt()     # Square root
?abs()          # Absolute value
?sin()          # Trig functions, also includes cos, tan, asin, atan, and atan2
?min()          # Minimum and maximum value of a vector
?max()
?which.min()    # Index of the min or max value of a vector
?which.max()
```

Math and Simulation

```
### Math Functions
# R has any basic math function can you think of:
?pmin()                 # Element-wise min or max of several vectors
?pmax()
?sum()              # Sum of the elements of a vector
?prod()             # Product of the elements of a vector
?cumsum()               # Cumulative sum of the elements of a vector
?cumprod()              # Cumulative product of the elements of a vector
?round()            # Round to the closest integer
?floor()            # Round to the closest integer lower
?ceiling()          # Round to the closest integer higher
?factorial()        # Factorial function
```

Math and Simulation

1) Cumulative sums and products

```
x <- c(12,5,13)

cumsum(x)

 [1]  12  17  30

cumprod(x)

 [1]  12  60  780
```

Math and Simulation

2) Minima and maxima

(1) min() combines all of its arguments into a vector and returns the minimum value:

```
x <- c(1,5,6)

y <- c(2,3,2)

min(x,y)

[1]  1
```

(2) pmin() compares these element-wise:

```
pmin(x,y)

[1]  1  3  2
```

Math and Simulation

2) Minima and maxima

(3) Minimum of a function

```
mytrig <- function(x)
{
    return(x^2-sin(x))
}
```

Math and Simulation

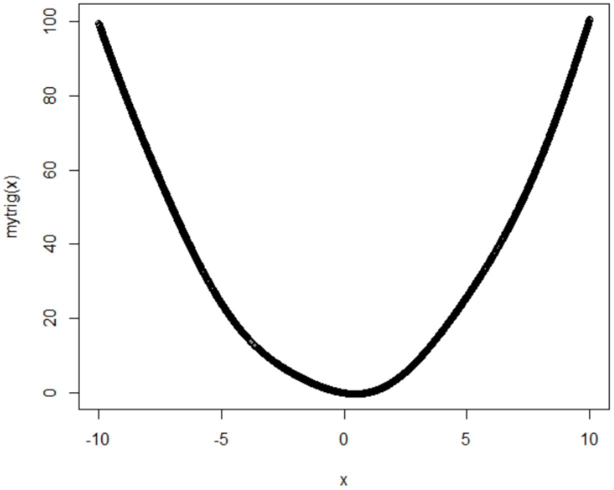
(3) Minimum of a function

Solution 1: generate a whole sequence of x values:

```
x <- seq(-10,10,by=0.01)

plot(x,mytrig(x))

min(mytrig(x))
```



Two issues here:

1) we are making a wild guess that the minimum falls within the range of the from and to values

2) we are assuming that an x precision of 0.01 is adequate to characterize the minimum property

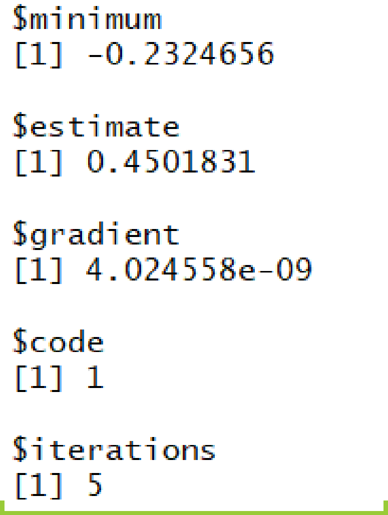
Math and Simulation

(3) Minimum of a function

Solution 2: Non-linear minimization of a function using a Newton-type algorithm:

nlm(mytrig,8)

# The second parameter is an initialization parameter (where to start looking).



```
$minimum
[1] -0.2324656

$estimate
[1] 0.4501831

$gradient
[1] 4.024558e-09

$code
[1] 1

$iterations
[1] 5
```

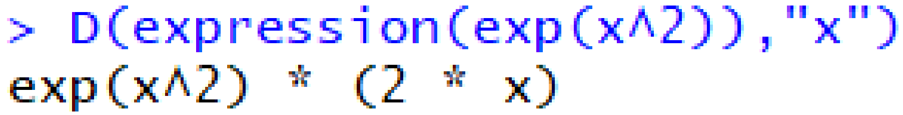
Math and Simulation

2) Calculus

One good package for solving differential equations is:

install.packages("odesolve")

(1) Derivative of a function: D()



```
> D(expression(exp(x^2)),"x")
exp(x^2) * (2 * x)
```

(2) Integration of a function: integrate()

integrate(function(x) x^2,lower=0,upper=1)

# You have to define the bounds

0.3333333

Math and Simulation

2) Functions for statistical distributions

R uses the following nomenclature:

- d: density/probability mass function (pmf)
- p: cumulative distribution function (cdf)
- q: quantiles
- r: random number generations.

Math and Simulation

2) Functions for statistical distributions

R uses the following nomenclature:

For a normal distribution:

?dnorm # Calculates the probability mass function of a normal distribution

?pnorm # Calculates the cumulative distribution function of a normal distribution

?qnorm # Calculates the quantile function of a normal distribution

?rnorm # Generates a set of random numbers drawn from a normal distribution.

Math and Simulation

2) Functions for statistical distributions

Let's look at a chi-square example:

```
mean(rchisq(1000,df=2))

# This pulls 1000 random values from a chi-square distribution with degrees of freedom equal to 2, and then calculates the mean value of these.

# What about the 95th percentile of the same distribution:

qchisq(p=0.95,df=2)
```

Or the 50th and the 95th at the same time:

```
qchisq(p=c(0.5,0.95),df=2)
```

Math and Simulation

3) Linear algebra operations on vectors and matrices

(1) Calculate the dot product (aka "inner product") of two vectors

```
crossprod(1:3,c(5,12,13))
```

(2) Matrix multiplication uses %*%:

```
a <- matrix(c(1,3,2,4),nrow=2)

b <- matrix(c(1,0,-1,1),nrow=2)

a %*% b
```

Math and Simulation

3) Linear algebra operations on vectors and matrices

(3) Solve a linear equation

x1 + x2 = 2

-x1 + x2 = 4

In matrix form, this is:

```
   /  1    1   \   / x1 \  =   / 2 \
   \ -1    1   /   \ x2 /      \ 4 /
```

```
a <- matrix(c(1,-1,1,1),nrow=2,ncol=2)
b <- c(2,4)
solve(a,b)
```

Math and Simulation

3) Linear algebra operations on vectors and matrices

(4) Set operations

```
x <- c(1,2,5)
y <- c(5,1,8,9)

# Union
union(x,y) # Notice duplicated values are gone.

# Intersect
intersect(x,y) # Only values shared between both sets.
```

Math and Simulation

3) Linear algebra operations on vectors and matrices

(4) Set operations

```
x <- c(1,2,5)
y <- c(5,1,8,9)

# setdiff: return values in x that are not in y:
setdiff(x,y)

# setequal: test for equality between sets:
setequal(x,c(1,2,5))
setequal(x,y)
```

Math and Simulation

3) Linear algebra operations on vectors and matrices

(4) Set operations

Build up more complex set operations. Determine all elements belonging to only one of the two sets:

```
symdiff <- function(a,b)
{
      sdfxy <- setdiff(x,y)
      sdfyx <- setdiff(y,x)
      return(union(sdfxy,sdfyx))
}
symdiff(x,y)
```

Math and Simulation

3) Linear algebra operations on vectors and matrices

(5) Random number generator?

rbinom() # Random numbers from a binomial distribution.

rnorm()   # Random numbers from a normal distribution.

runif()   # Random numbers from a uniform distribution.

Math and Simulation

3) Linear algebra operations on vectors and matrices

(5) Random number generator

Recreate a random number stream by setting the seed. The seed is an initialization parameter to the "random" number generator.

```
set.seed(1234)

runif(10)

set.seed(1234)

runif(10)
```

Quiz 4

Generate a vector of X that has 100 random numbers from the normal distribution (mean = 1, sd = 1).

Generate a vector of Y that has 200 random numbers from the uniform distribution (min = 0, max = 2)

Then generate a list Z with two elements X and Y, and use lapply() to count the number of values between 0.5 and 1.5 for each element.

The R file needs to be named:

LastName_FirstName_Quiz4.R

Please submit the quiz R file on Compass by the end of this class.