Week 1 - Donny Lofland

library(Deriv)  
library(mosaic)

## Registered S3 method overwritten by 'mosaic':  
## method from   
## fortify.SpatialPolygonsDataFrame ggplot2

##   
## The 'mosaic' package masks several functions from core packages in order to add   
## additional features. The original behavior of these functions should not be affected by this.

##   
## Attaching package: 'mosaic'

## The following objects are masked from 'package:dplyr':  
##   
## count, do, tally

## The following object is masked from 'package:Matrix':  
##   
## mean

## The following object is masked from 'package:ggplot2':  
##   
## stat

## The following objects are masked from 'package:stats':  
##   
## binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,  
## quantile, sd, t.test, var

## The following objects are masked from 'package:base':  
##   
## max, mean, min, prod, range, sample, sum

library(mosaicCalc)

## Loading required package: mosaicCore

##   
## Attaching package: 'mosaicCore'

## The following objects are masked from 'package:dplyr':  
##   
## count, tally

##   
## Attaching package: 'mosaicCalc'

## The following object is masked from 'package:stats':  
##   
## D

library(rSymPy)

## Loading required package: rJython

## Loading required package: rJava

## Loading required package: rjson

##   
## Attaching package: 'rSymPy'

## The following object is masked from 'package:Matrix':  
##   
## Matrix

Find the derivatives with the respect to x of the following.

1. F(x|x≥0)=

myf1=function(x){1-exp(-lambda\*x)}  
print(Deriv(myf1))

## function (x)   
## lambda \* exp(-(lambda \* x))

myf2 = function(x){(x-a)/(b-a)}  
print(Deriv(myf2))

## function (x)   
## 1/(b - a)

myf3 = function(x){(x-a)^2/((b-a)(c-a))}  
print(Deriv(myf3))

## Warning in if (che1 == "stop") {: the condition has length > 1 and only the  
## first element will be used

## Warning in if (che1 != "[") {: the condition has length > 1 and only the first  
## element will be used

## Warning in if (stch == "function") {: the condition has length > 1 and only the  
## first element will be used

## function (x)   
## 2 \* ((x - a)/(b - a)(c - a))

myf4 = function(x)1-(b-x)^2/((b-a)(c-a))  
print(Deriv(myf4))

## Warning in if (che1 == "stop") {: the condition has length > 1 and only the  
## first element will be used

## Warning in if (che1 != "[") {: the condition has length > 1 and only the first  
## element will be used

## Warning in if (stch == "function") {: the condition has length > 1 and only the  
## first element will be used

## function (x)   
## 2 \* ((b - x)/(b - a)(c - a))

Solve the following definite and indefinite integrals

myf5=function(x)3\*x^3  
integrate(Vectorize(myf5),0,10)

## 7500 with absolute error < 8.3e-11

Solution:

Substitution ; ; ;

;

solve by Parts: ; ;

formula:

plugging back in:

replacing back in:

lastly we need to integrate over :

library(rSymPy)  
sympy("x = Symbol('x')")

## [1] "x"

sympy("l = Symbol('lambda')")

## [1] "lambda"

sympy("integrate(x\*l\*exp(-l\*x),(x,0,x))")

## [1] "lambda\*(-exp(-lambda\*x)/lambda\*\*2 - x\*exp(-lambda\*x)/lambda) + 1/lambda"

Answer from sympy:

I’m not sure how to get r to calculate this or it was intended that we to this by hand. I made my best stab at doing by hand and code in latex for the notebook.

Solution:

library(rSymPy)  
sympy("x = Symbol('x')")

## [1] "x"

sympy("a = Symbol('alpha')")

## [1] "alpha"

sympy("B = Symbol('beta')")

## [1] "beta"

sympy("integrate(1/(B-a),(x,0,x))")

## [1] "x/(beta - alpha)"

I’m not sure how to get r to calculate this or it was intended that we to this by hand. I’m a python guy so brought over sympy to help. I made my best stab at doing by hand and code in latex for the notebook.

Solution:

library(rSymPy)  
sympy("x = Symbol('x')")

## [1] "x"

sympy("a = Symbol('alpha')")

## [1] "alpha"

sympy("B = Symbol('beta')")

## [1] "beta"

sympy("G = Symbol('Gamma')")

## [1] "Gamma"

sympy("integrate(x \* (G\*a\*B\*\*a)\*\*(-1) \* x\*\*(a) \* exp(-B\*x), (x,0,x))")

## [1] "Integral(x\*beta\*\*(-alpha)\*x\*\*alpha\*exp(-beta\*x)/(Gamma\*alpha), (x, 0, x))"

sympy solution:

I really have no idea how to tackle this or was was hinted by the gamma function. I’m hoping we can cover this at some point so I can understand exactly how to approach the problem. Did you want us to hand solve this, use r or some combination? I had assumed this was a r task since we were using R to solve the early problems. Anyways, for problem 7 & 8, I could certainly use some insights. Thanks! Donny

Hint: the last part of the equation is beginning with the gamma function is a Gamma probability distribution function. Try rearranging the terms to integrate another Gamma distribution out of the integral, as pdfs must integrate to 1.

With the following matrix,

myMatrix <- matrix(c(1,3,4,2,3,6,3,1,8),3,3)  
myMatrix

## [,1] [,2] [,3]  
## [1,] 1 2 3  
## [2,] 3 3 1  
## [3,] 4 6 8

1. Invert it using Gaussian row reduction.

library(matlib)

##   
## Attaching package: 'matlib'

## The following object is masked from 'package:rJava':  
##   
## J

gaussianElimination(myMatrix, numeric(3))

## [,1] [,2] [,3] [,4]  
## [1,] 1 0 0 0  
## [2,] 0 1 0 0  
## [3,] 0 0 1 0

gaussianElimination(myMatrix, diag(3))

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 1 0 0 -4.5 -0.5 1.75  
## [2,] 0 1 0 5.0 1.0 -2.00  
## [3,] 0 0 1 -1.5 -0.5 0.75

inv(myMatrix)

## [,1] [,2] [,3]  
## [1,] -4.5 -0.5 1.75  
## [2,] 5.0 1.0 -2.00  
## [3,] -1.5 -0.5 0.75

1. Find the determinant.

det(myMatrix)

## [1] -4

1. Conduct LU decomposition

lum <- lu(myMatrix)  
elu <- expand(lum)  
  
(L <- elu$L)

## 3 x 3 Matrix of class "dtrMatrix" (unitriangular)  
## [,1] [,2] [,3]   
## [1,] 1.0000000 . .  
## [2,] 0.7500000 1.0000000 .  
## [3,] 0.2500000 -0.3333333 1.0000000

(U <- elu$U)

## 3 x 3 Matrix of class "dtrMatrix"  
## [,1] [,2] [,3]   
## [1,] 4.0000000 6.0000000 8.0000000  
## [2,] . -1.5000000 -5.0000000  
## [3,] . . -0.6666667

(P <- elu$P)

## 3 x 3 sparse Matrix of class "pMatrix"  
##   
## [1,] . . |  
## [2,] . | .  
## [3,] | . .

L %\*% U

## 3 x 3 Matrix of class "dgeMatrix"  
## [,1] [,2] [,3]  
## [1,] 4 6 8  
## [2,] 3 3 1  
## [3,] 1 2 3

1. Multiply the matrix by it’s inverse.

inv <- round(solve(myMatrix),2)  
myMatrix%\*%inv

## [,1] [,2] [,3]  
## [1,] 1 0 0  
## [2,] 0 1 0  
## [3,] 0 0 1