732A38 Computational statistics

Hassaan Tahir

***Assignment 3: Variance***

1. Write your own function *myvar* estimating variance in this way

myvar<-function(x)

{

n<-length(x)

sumx2 <-sum(x)\*sum(x);

sumsq<-sum(x\*x);

v<-(1/(n-1)\*(sumsq-sumx2/n));

y=v-var(x);

y

}

1. Generate vector x with 10000 random numbers, normally distributed with mean 108 and variance 1.

x=rnorm(10000, mean=1e8, sd=1)

1. For each subset Xi= {x1…xi}, i=1…10000 compute difference Yi= myvar(x)-var(x), where var(x) is a standard variance estimation function in R. Plot the dependence Yi on Xi. Draw necessary conclusions. How well your function works? What is the reason behind such behavior?

I could not run the loop but I have checked it for different numbers of observations. . when we will increase the number of observations the difference Yi= myvar(x)-var(x), where var(x) will increase because of the rounding of error, the variance myvar will not give very accurate results for large number of observations. R uses another formula to calculate variance OK- (plot is missing)

***Assignment 2: Derivative***

1. Write your own function computing the derivative of function *f(x)=*|*x*| in this way. Take ε=10-15

**Code:**

>fun<-function(x)

+ {

+ f3<-abs(x)

+ e<-.000000000000001

+ f4<-abs(x+e)

+ d<- (f4-f3)/e

+ d

+ }

1. Compute your derivative function at point x=100000.

> fun(100000)

[1] 0

1. What is the value you obtained? What is the real value of the derivative? Explain the reason behind the discovered difference

The value I have obtained from function is zero and the real value of function is 1.The reason for this is that the value of epsilon is very small. If we add any big number like “100000”, in it, the result will be “100000” because of the computer rounded off this value and neglect the value of epsilon. When we subtract f4 (which is now “100000”) from f3 (which was already “100000”) the result will be “0”. Like as under.

> x<- 10000;

> y<- 0.000000000000001;

> z<-x+y;

> z

[1] 10000

Now our numerator becomes zero and what ever the denumerators of the value, zero numerators gives the value zero in dividing. OK

***Assignment 1: Be careful with ‘==’***

1. Check the result of this program. Comment why this happened.

The result of this program is “Teacher lied”. Because of rounding off or a very small inaccuracy of the floating point integers the result is like this. As’==’ mean exact equality. The basic principle is this: computers don't store numbers (except smallish integers and some fractions) exactly. E.g. result of 1/3 is stored as a number close to, but not exactly, one third. Therefore we  **should not test floating point numbers for exact equality.**

Specify how the program can be modified to give a correct result

Instead of checking for exact equality, we can modify our program as follows

>x1<-1/3;

> x2<-1/4;

> if (x1-x2-1/12<.000000000000001){

+ print("Teacher said true")

+ } else{

+ print("Teacher lied")}

[1] "Teacher said true" OK Almost correct. abs(x1-x2-1/12)<.000000000000001 is better