Calculate the Area under a Curve

I would like to calculate the area under a curve to do integration without defining a function such as in `integrate()` .

My data looks as this:

```
Date          Strike     Volatility
2003-01-01    20         0.2
2003-01-01    30         0.3
2003-01-01    40         0.4
etc.
```

I plotted `plot(strike, volatility)` to look at the volatility smile. Is there a way to integrate this plotted "curve"?

r numerical-integration

edited Oct 12 '17 at 19:03

Victor Klos
146 3 10

asked Feb 10 '11 at 7:35

Dani
747 4 11 17

1 Have a look at this related question: stackoverflow.com/questions/4903092/calculate-auc-in-r – Andrie Feb 10 '11 at 8:40

3 @Andrie : that's a different type of AUC... – Joris Meys Feb 10 '11 at 9:07

7 Answers

The AUC is approximated pretty easily by looking at a lot of trapezium figures, each time bound between `x_i` , `x_{i+1}` , `y{i+1}` and `y_i` . Using the rollmean of the zoo package, you can do:

```
library(zoo)
```

```
AUC <- sum(diff(x[id])*rollmean(y[id],2))
```

Make sure you order the x values, or your outcome won't make sense. If you have negative values somewhere along the y axis, you'd have to figure out how exactly you want to define the area under the curve, and adjust accordingly (e.g. using `abs()`)

Regarding your follow-up : if you don't have a formal function, how would you plot it? So if you only have values, the only thing you can approximate is a definite integral. Even if you have the function in R, you can only calculate definite integrals using `integrate()` . Plotting the formal function is only possible if you can also define it.

edited Jun 14 '17 at 12:55

Sotos
21.8k 5 14 35

answered Feb 10 '11 at 9:07

Joris Meys
71.8k 18 167 230

Thanks, that works. Is there also a way to plot the integral? I mean, if I have a curve such as the volatility smile, I should be able to plot its integral which is a curve as well. – Dani Feb 10 '11 at 10:24

Thats a great way to test that my pdfs sum to 1. Thanks! – David LeBauer Mar 29 '11 at 22:32

This is great, but if some values are missing, the formula won't work anymore. – Dan Chaltiel Jun 14 '17 at 15:47

@DanChaltiel if some values are missing, there's no way of knowing what the real area under the curve is. So that seems not a problem to me. Just remove the missing observations before calculation if you want to disregard the missing data. – Joris Meys Jun 14 '17 at 16:01

@JorisMeys If you have 10 x values and only 9 y values, you can have a pretty good approximation of AUC if you don't count the missing value. Removing all sample that have only one NA seems a waste for me. – Dan Chaltiel Jun 15 '17 at 8:03

Just add the following to your program and you will get the area under the curve:

```
require(pracma)
AUC = trapz(strike,volatility)
```

From `?trapz` :

> This approach matches exactly the approximation for integrating the function using the trapezoidal rule with basepoints x.

edited Dec 7 '12 at 20:53
Ari B. Friedman

answered Mar 1 '12 at 21:08
simon

 **43.3k** 19 140 198  **271** 3 2

1 Details are always welcome, especially when an answer has already been accepted. – Nikana Reklawyks Oct 26 '12 at 7:40

Be advised that `trapz()` will give you a negative value if your `x` values are decreasing. See `x<-1:10` vs `x<-10:1` . – Matt Jul 7 '16 at 13:41

Three more options, including one using a spline method and one using Simpson's rule...

```
# get data
n <- 100
mean <- 50
sd <- 50

x <- seq(20, 80, length=n)
y <- dnorm(x, mean, sd) *100

# using sintegral in Bolstad2
require(Bolstad2)
sintegral(x,y)$int

# using auc in MESS
require(MESS)
auc(x,y, type = 'spline')

# using integrate.xy in sfsmisc
require(sfsmisc)
integrate.xy(x,y)
```

The trapezoidal method is less accurate than the spline method, so `MESS::auc` (uses spline method) or `Bolstad2::sintegral` (uses Simpson's rule) should probably be preferred. DIY versions of these (and an additional approach using the quadrature rule) are here: http://www.r-bloggers.com/one-dimensional-integrals/

edited Jan 30 '13 at 8:03

answered Jan 29 '13 at 21:34

 Ben
27.7k 9 81 152

1 There is another package called "flux". It has the same function name that "MESS" has, "auc()". It worth a try! – Facottons Mar 16 '17 at 21:16

OK so I arrive a bit late at the party but going over the answers a plain `R` solution to the

problem is missing. Here goes, simple and clean:

```
sum(diff(x) * (head(y,-1)+tail(y,-1)))/2
```

The solution for OP then reads as:

```
sum(diff(strike) * (head(volatility,-1)+tail(volatility,-1)))/2
```

This effectively calculates the area using the trapezoidal method by taking the average of the "left" and "right" y-values.

NB: as @Joris already pointed out you could use `abs(y)` if that would make more sense.

edited May 17 '15 at 11:47

answered May 16 '15 at 21:19

Victor Klos
146 3 10

I always prefer plain `R` solutions :) – Verbal Jan 14 '17 at 17:53

In the pharmacokinetics (PK) world, calculating different types of AUC is a common and fundamental task. The are lots of different AUC calculations for pharmacokietics, such as

- AUC0-t = AUC from zero to time t
- AUC0-last = AUC from zero to the last time point (may be same as above)
- AUC0-inf = AUC from zero to time infinity
- AUCint = AUC over a time interval
- AUCall = AUC over the whole time period for which data exists

One of the best packages which does these calculations is the relatively new package `PKNCA` from the folks at Pfizer. Check it out.

answered May 3 '16 at 20:42

hackR
726 5 17

Joris Meys's answer was great but I struggled to remove NAs from my samples. Here is the little function I wrote to deal with them :

```
library(zoo) #for the rollmean function

######
#' Calculate the Area Under Curve of y~x
#'
#'@param y Your y values (measures ?)
#'@param x Your x values (time ?)
#'@param start : The first x value
#'@param stop : The last x value
#'@param na.stop : returns NA if one value is NA
#'@param ex.na.stop : returns NA if the first or the last value is NA
#'
#'@examples
#'myX = 1:5
#'myY = c(17, 25, NA, 35, 56)
#'auc(myY, myX)
#'auc(myY, myX, na.stop=TRUE)
#'myY = c(17, 25, 28, 35, NA)
#'auc(myY, myX, ex.na.stop=FALSE)
auc = function(y, x, start=first(x), stop=last(x), na.stop=FALSE, ex.na.stop=TRUE){
  if(all(is.na(y))) return(NA)
  bounds = which(x==start):which(x==stop)
  x=x[bounds]
  y=y[bounds]
  r = which(is.na(y))
  if(length(r)>0){
    if(na.stop==TRUE) return(NA)
    if(ex.na.stop==TRUE & (is.na(first(y)) | is.na(last(y)))) return(NA)
    if(is.na(last(y))) warning("Last value is NA, so this AUC is bad and you should feel
bad", call. = FALSE)
    if(is.na(first(y))) warning("First value is NA, so this AUC is bad and you should feel
bad", call. = FALSE)
    x = x[-r]
    y = y[-r]
  }
  sum(diff(x[order(x)])*rollmean(y[order(x)],2))
}
```

I then use it with an apply onto my dataframe : `myDF$auc = apply(myDF, MARGIN=1, FUN=auc, x=c(0,5,10,15,20))`

Hope it can help noobs like me :-)

EDIT : added bounds

edited Jun 15 '17 at 12:00

answered Jun 15 '17 at 9:53

Dan Chaltiel

1,059 1 14 27

You can use ROCR package, where the following lines will give you the AUC:

```
pred <- prediction(classifier.labels, actual.labs)
attributes(performance(pred, 'auc'))$y.values[[1]]
```

answered Oct 26 '12 at 8:26

rezakhorshidi
489 1 4 8

4 The OP doesn't want to compute ROC curve and its AUC, but the area under an arbitrary curve. – Calimo Mar 14 '14 at 11:19