AUC Examples

David Elsheimer

2/26/2020

Monte Carlo

The following is sample code showing estimation of area under the curve for a standard normal distribution.

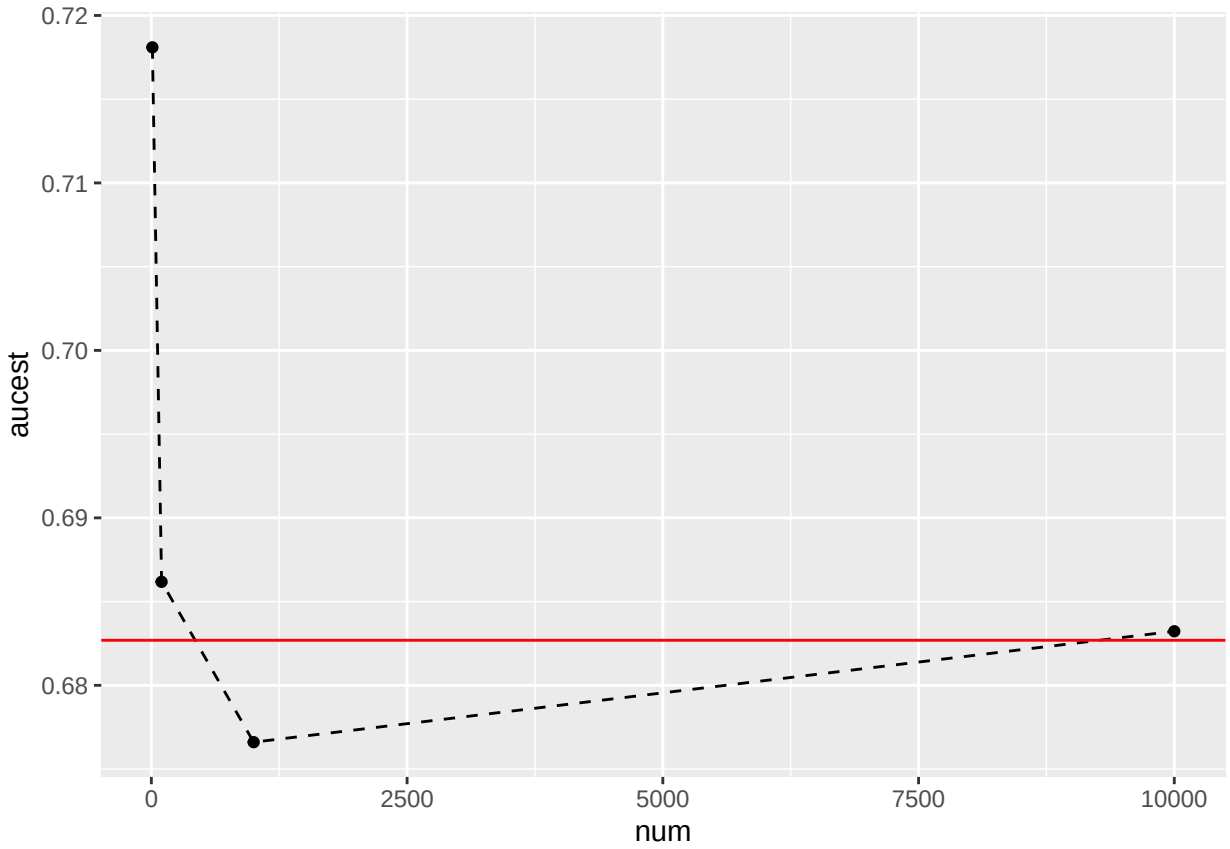
Note the true value of $AUC(-1,1) = \int_a^b \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \Phi(1) - \Phi(-1)$. This can easily be calculated in R using `pnorm`. The true value is found to be 0.6826895.

A vector of values of the estimated AUC for $n \in \{10, 100, 1000, 10000\}$ is provided below.

```
library(ggplot2)
set.seed(1002)
aucest<- c(RStudio2020::estimate_auc(10,-1,1),
RStudio2020::estimate_auc(1e2,-1,1),
RStudio2020::estimate_auc(1e3,-1,1),
RStudio2020::estimate_auc(1e4,-1,1))
num<- c(10,100,1000,10000)
plotdata<- as.data.frame(cbind(num,aucest))
plotdata
```

```
##     num    aucest
## 1    10 0.7180961
## 2   100 0.6861807
## 3  1000 0.6766061
## 4 10000 0.6832285
```

```
ggplot(data = plotdata, aes(x=num, y=aucest)) + geom_point()+
    geom_line(linetype="dashed")+geom_hline(yintercept=0.6826895, color = "red")
```



Let $Z_1, Z_2, ..., Z_n$ denote n i.i.d random variables such that $|Z_i| \leq 1$ for $i = 1, ..., n$. Let $\overline{Z}$ denote the sample mean of Z_i and μ denote the mean of Z_1. Recall that the Hoeffding bound tell us that

$$P(|\overline{Z} - \mu| \geq t) \leq 2\exp(-2nt^2).$$

Here, $Z_i = I(Y_i \leq f(X_i))$. The Z_i's are i.i.d., and $|Z_i| \leq 1$. Thus a Hoeffding bound is applicable here. $\bar{Z} = \frac{1}{n}\sum_{i=1}^{n} Z_i$. Here $E(Z_i) = P(Y_i \leq f(X_i))$. Due to Y_i's being i.i.d, as well as X_i's being i.i.d., this can be further generalized as $P(Y \leq f(X))$.

Further note that $P(Y \leq f(X)) = \frac{S}{c(b-a)}$, where a, b are the bounds for X_i, and c is the maximum density of $f(X)$, and S is the area under the curve.

Therefore, $S = c(b-a)P(Y \leq f(X)) \approx c(b-a)\bar{Z}$.

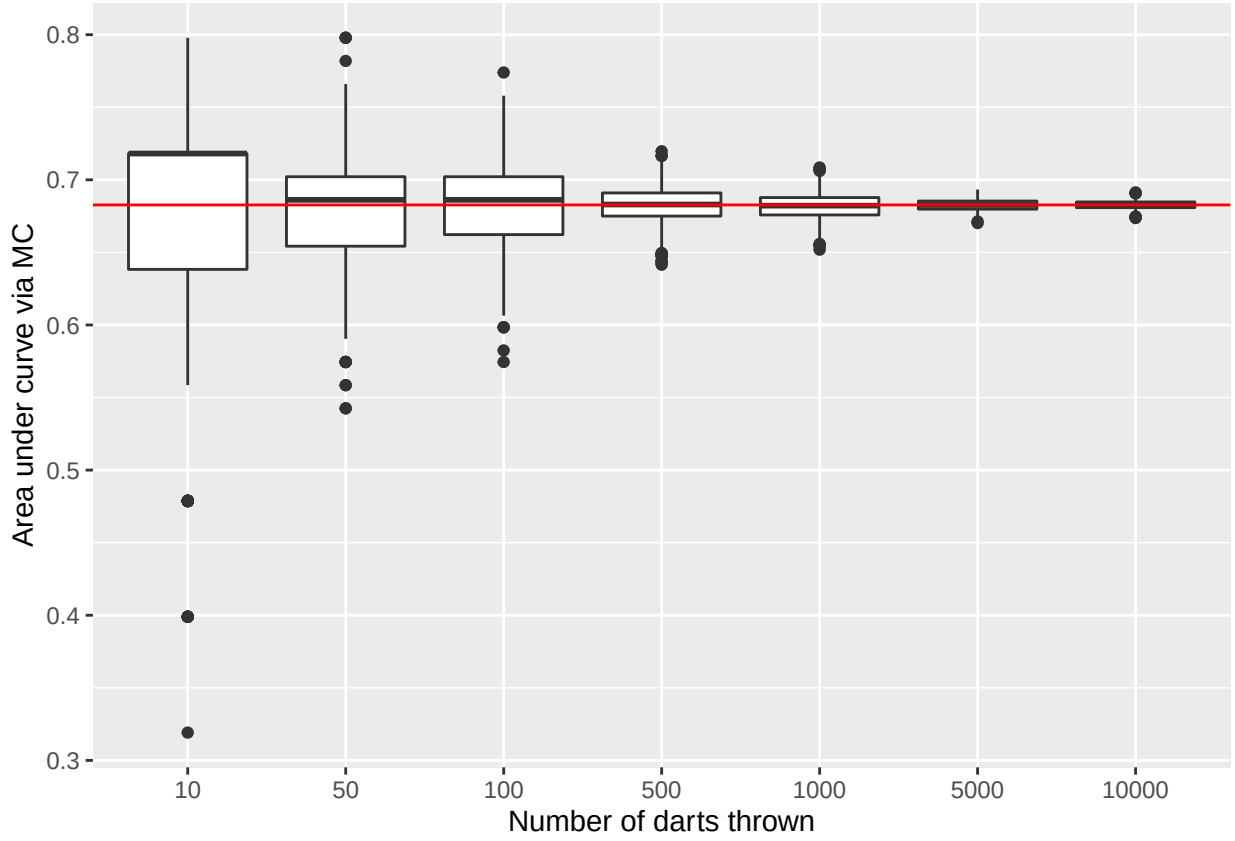
Applying the Hoeffding bound, $P(|c(b-a)\bar{Z} - c(b-a)P(Y \leq f(X))| \geq c(b-a)t) \leq 2\exp(-2n(c(b-a)t)^2)$.

Applying what has been demonstrated above, we can assess the correctness of `estimate_auc` using the following R script.

```
library(tidyr)
library(dplyr)
library(forcats)
library(ggplot2)
set.seed(1002)

#Simulating estimates for different levels of n
#Creating boxplots at each level
num<- c(10,50,100,500,1000,5000,10000)
```

```
auc_eval <- replicate(sapply(num,a=-1,b=1, RStudio2020::estimate_auc), n =1000)
auc_data <- as.data.frame(t(auc_eval))
colnames(auc_data) <- c("10", "50", "100", "500","1000","5000","10000")
longauc <- gather(auc_data,key=num)
longauc$num<-factor(longauc$num, levels=c("10", "50", "100", "500","1000","5000","10000"))
ggplot(data = longauc, aes(x=num, y=value)) + geom_boxplot() +
    geom_hline(yintercept=0.6826895, color = "red") +
  xlab("Number of darts thrown") +
  ylab("Area under curve via MC")
```




```
#Checking that the probablities will be less than the bounds
probbound<-function(n,t){
  2*exp(-2*n*t**2)
}

true_y <- pnorm(1)-pnorm(-1)
longauc$count_tol1 <- 1- as.numeric(between(longauc$value,true_y-0.1,true_y+0.1))
longauc$count_tol2 <- 1-as.numeric(between(longauc$value,true_y-0.05,true_y+0.05))
longauc$count_tol3 <- 1-as.numeric(between(longauc$value,true_y-0.025,true_y+0.025))
probapprox <- longauc %>% group_by(num)%>%
  summarise(mean(count_tol1), mean(count_tol2),mean(count_tol3))
orderedprob <- probapprox
orderedprob$probtol1 <-sapply(num, t=0.1, probbound)
orderedprob$probtol2 <-sapply(num, t=0.05, probbound)
orderedprob$probtol3 <- sapply(num, t=0.025, probbound)
knitr::kable(orderedprob, col.names = c("N", "$\\hat{P}(est\\notin$ tol-int 1)",
```

```
                          " $\\hat{P}(est\\notin$ tol-int 2)",
                          "$\\hat{P}(est\\notin$ tol-int 3)",
                          "Tol1 True prob",
                          "Tol2 True prob",
                          "Tol3 True prob"))
```

N	$\hat{P}(est \notin$ tol-int 1)	$\hat{P}(est \notin$ tol-int 2)	$\hat{P}(est \notin$ tol-int 3)	Tol1 True prob	Tol2 True prob	Tol3 True prob
10	0.367	0.367	1.000	1.6374615	1.9024588	1.9751556
50	0.013	0.236	0.523	0.7357589	1.5576016	1.8788261
100	0.002	0.093	0.386	0.2706706	1.2130613	1.7649938
500	0.000	0.000	0.043	0.0000908	0.1641700	1.0705229
1000	0.000	0.000	0.007	0.0000000	0.0134759	0.5730096
5000	0.000	0.000	0.000	0.0000000	0.0000000	0.0038609
10000	0.000	0.000	0.000	0.0000000	0.0000000	0.0000075

A modified AUC function was also created which returns an estimated AUC with a specified tolerance of probability `p`.

```
RStudio2020::estimate_auc_tol(-1,1,0.5,0.5)
```

```
## [1] 0.287776 3.000000
```

```
RStudio2020::estimate_auc_tol(-1,1,1,0.1)
```

```
## [1] 0 2
```

```
RStudio2020::estimate_auc_tol(-1,1,0.05,0.05)
```

```
## [1]   0.6834908 738.0000000
```

As the probability decreases, the estimate requires larger and larger n, and the estimate trends towards the true AUC.

Suppose instead of the Y-space is changed, such that the spac eis now $[a,b] \times [0,c]$ for $c \geq 1/\sqrt{2\pi}$. With the fraction of points under the curve still being what is estimated.

The modification is simple. The change here is that $Y \sim U(0,c)$. This will change the estimator with respect to how the height of the canvas changes. Notice that for fixed a, b where $b > a$, X is bounded between $(\frac{1}{\sqrt{2\pi}}e^{-b^2/2}, \frac{1}{\sqrt{2\pi}})$. As c increases, $P(A_1) \approx \frac{1}{n}\sum_{i=1}^{n} I(A_i)$ will decrease because it will become increasingly likely that there will be values of $y_i > f(x_i)$. Thus the quality of the estimator will decrease for greater values of c. This can be coded fairly easily, as seen above and in the ackage. A condition is included to return errors for canvases shorter than the minimum height if $\frac{1}{\sqrt{2\pi}}$.

```
set.seed(1002)
#demonstrating the behavior
RStudio2020::estimate_auc_modified(10,-1,1, c= .1)
```

```
## Error in RStudio2020::estimate_auc_modified(10, -1, 1, c = 0.1): Canvas height too low
```

```
RStudio2020::estimate_auc_modified(10,-1,1)
```

```
## [1] 0.7180961
```

```
RStudio2020::estimate_auc_modified(10,-1,1, c= 3)
```

```
## [1] 0.07978846
```

```
RStudio2020::estimate_auc_modified(10,-1,1, c= 4)
```

```
## [1] 0
```